POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Analisi di dati spazio-temporali mediante tecniche di Data Mining

Estrazione e visualizzazione di regole di associazione per l'analisi di dati spazio-temporali

Relatore:
prof. Paolo Garza

Candidato:
Federico Ferrero

Anno accademico 2018-2019

Sommario

Il candidato propone un approccio e uno studio generale sull'analisi, nell'ambito del campo dei Big Data, di dati spazio-temporali. In particolare, nel lavoro presentato, vengono proposti un metodo per estrarre regole di associazione relative a eventi georeferenziati, per capire come un evento che si verifica in un determinato luogo è collegato ad un altro evento che si verifica in un altro luogo, e alcune modalità di rappresentazione grafica dei risultati ottenuti. Questo metodo può essere utilizzato per elaborare delle previsioni relative allo stato di alcuni punti georeferenziati e, nel caso studio presente all'interno del lavoro di tesi, è stato testato su un dataset relativo allo stato di alcune stazioni di biciclette presenti nella città di Barcellona.

Ringraziamenti

Il candidato ringrazia vivamente la propria famiglia, fidanzata ed amici per il supporto morale fornito durante lo svolgimento della tesi.

Un doveroso grazie ad Enrico Panetta per le spiegazioni relative all'utilizzo del Cluster del Politecnico e i comandi da utilizzare per poterne vedere i risultati.

Un ringraziamento particolare al professor Paolo Garza per l'aiuto, la disponibilità e i consigli messi a disposizione durante i mesi di lavoro.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Il lavoro descritto nel seguente documento spiega sia le basi e i concetti utilizzati per lo sviluppo del lavoro che l'approccio utilizzato per affrontare il problema, concludendo con l'analisi di un caso studio e i risultati osservati e ottenuti.

Il secondo capitolo contiene un'analisi generale sullo stato dell'arte relativo allo studio di dati spazio-temporali, cominciando da una descrizione di cosa sono i big data e analizzando quali esperimenti e tecniche vengono usate per elaborarli. In particolare viene descritto cosa sono le regole di associazione e gli itemset insieme ai due principali algoritmi che permettono di estrarre questi ultimi (algoritmo Apriori e FPGrowth). Dopo di questo vengono riportate alcune definizioni formali applicate all'ambito dei dati spazio-temporali e illustrati due esempi di studi effettuati negli ultimi anni; il primo incentrato sull'unione di dati spaziali e non, il secondo invece più improntato sulle tecniche di visualizzazione di questo tipo di informazioni e dei risultati ottenuti dopo averle elaborate.

Il terzo capitolo descrive l'approccio e il metodo utilizzato per risolvere il problema proposto. Per prima cosa viene spiegata l'idea alla base dell'algoritmo analizzato durante il lavoro di tesi, valido per qualsiasi tipo di informazione spazio-temporale espressa in un modo che verrà descritto successivamente, e le varie elaborazioni e trasformazioni dei dati che lo compongono; in secondo luogo vengono mostrate le modalità pensate ed implementate per la visualizzazione dei risultati e delle regole estratte, con relative immagini ed esempi grafici che spiegano e illustrano come le applicazioni sono state realizzate.

Il quarto capitolo è dedicato alle osservazioni e agli esperimenti condotti dopo la realizzazione delle applicazioni di cui si è appena discusso. Tramite grafici e diagrammi dettagliati viene illustrato e spiegato come la variazione di alcuni parametri dell'algoritmo pensato influisce sulle performance, sui risultati e sulle regole estratte. Ogni parametro viene analizzato a sè in modo da avere un'idea precisa di come esso si relaziona con gli altri. In queste sezioni è anche presente un'analisi sull'efficienza

delle regole estratte nel caso esse vengano utilizzate per effettuare delle predizioni, con tabelle riepilogative che illustrano la precisione dei risultati ottenuti e la correttezza delle predizioni.

L'ultimo capitolo, infine, è dedicato alle conclusioni del lavoro realizzato. Esso illustra le informazioni che si possono dedurre dai risultati ottenuti ed elenca alcune possibili migliorie effettuabili sia per l'applicazione dedicata al processamento dei dati e all'estrazione delle regole, sia per quella sviluppata invece per la visualizzazione dei risultati.

Capitolo 2

Stato dell'arte

2.1 Data mining per dati spazio-temporali

Il lavoro di tesi realizzato e presentato in questo documento è stato svolto nel dominio dell'analisi dati e, in particolare, nell'ambito dell'estrazione di regole di associazione e itemset. Il mondo dei big data è un mondo che negli ultimi anni ha suscitato, sia nell'ambito della ricerca che in quello aziendale, un interesse sempre maggiore, arrivando a diventare uno dei campi più richiesti sul mercato. Questo perché ormai, grazie alla tecnologia sempre maggiore, siamo circondati da una quantità di dati enorme che risulta sempre più difficile da analizzare e comprendere. Sono molti gli studi che hanno dimostrato che nei prossimi anni il numero di dati a nostra disposizione crescerà in maniera esponenziale, arrivando a costruire una rete di informazioni utile e quasi indispensabile al miglioramento delle condizioni di vita. Si pensi per esempio alle smart cities (città intelligenti i cui dati sono analizzati in tempo reale e utilizzati per migliorare il benessere delle persone che le abitano) o alla possibilità di avere sistemi di guida automatizzati in base alle informazioni sul traffico in tempo reale. La necessità di estrapolare informazioni e lo stesso aumento e sviluppo della tecnologia hanno portato allo sviluppo e alla determinazione di un set di tecniche e metodi di analisi adatte proprio a questa enorme mole di dati, che vanno a formare quello che prende il nome di Data Mining.

Il vantaggio più grande di queste tecniche è che possono essere applicate a qualsiasi tipo di dato o dataset, permettendo potenzialmente l'analisi di qualsiasi fenomeno su larga scala. I dati georeferenziati o spazio-temporali, che sono il principale soggetto e oggetto di analisi di questa tesi, non fanno eccezione. Sono molteplici infatti gli studi e i casi analizzati riguardanti dati come precipitazioni, percorsi geografici e variazione di parametri nel tempo. Tutti questi studi si basano, ovviamente, sui concetti e gli algoritmi base del Data Mining, senza i quali non sarebbe possibile ottenere risultati su larga scala e in maniera efficiente.

Per questo motivo, prima di entrare nel dettaglio del lavoro sviluppato, è bene riassumere i concetti fondamentali che sono stati utilizzati, presentando anche alcuni esempi di studio che sono stati effettuati in ambito simile a quello della tesi presentata. In questo capitolo verrà dunque spiegato che cosa è un itemset e cosa

una regola di associazione, come essi vengono analizzati e i diversi modi che esistono per farlo.

2.2 Itemset e regole di associazione

Gli itemset e le regole di associazione sono definiti all'interno di dataset costituiti da un insieme di transazioni. Una transazione può essere descritta come un insieme di oggetti che compongono un record del dataset preso in esame. Prendendo per esempio un dataset relativo alla spesa di diversi clienti di un supermercato, ogni transazione può essere associata alla spesa di un singolo cliente, all'interno della quale gli item corrispondono ai singoli oggetti acquistati.

ID	item
1	pasta, pane, olio, sale, pepe
2	shampoo, pane, insalata, pomodori
3	pepe, origano
4	pepe, pane, olio
5	sale, prosciutto, biscotti
6	olio, prosciutto, pollo, pane, marmellata

Tabella 2.1. Esempio di semplice dataset contenente 6 transazioni

Nella tabella mostrata è possibile notare un esempio di dataset all'interno del quale ogni transazione è identificata da un id. Ogni transazione contiene un numero di item che può essere uguale o diverso a quello delle altre transazioni e, al suo interno, gli item non sono ordinati e nemmeno duplicati, questo perché ognuno di essi sia univoco e rappresentato solo una volta per la sua transazione di riferimento.

In particolare, si definisce itemset un insieme di item all'interno di una transazione. Esso può comprendere tutti gli item di una transazione o un suo sottoinsieme e, in questo ultimo caso, si definisce K-itemset un itemset che contiene K item. Prendendo ad esempio la transazione con ID 2 nella tabella di esempio potremmo dire che:

shampoo, pane, insalata è un 3-itemset

mentre:

insalata, pomodori

è un 2-itemset corrispondente alla stessa transazione.

Il concetto fondamentale legato agli itemset è quello di supporto. Con esso si intende il numero di transazioni che contengono un determinato itemset rispetto al numero di transazioni totali. Facendo riferimento alla tabella di esempio precedente:

$$sup(pane, olio) = \frac{3}{6}$$

Perché l'itemset (pane, olio) compare nelle transazioni 1, 4 e 6 rispetto a tutti i record presenti nella tabella. In questo contesto si definisce Frequent itemset un itemset il cui supporto è maggiore di una soglia prestabilita. Se si usasse come soglia 0.4 l'itemset (pane, olio) sarebbe un Frequent itemset perché il suo supporto vale 0.5, mentre se si scegliesse come soglia 0.6 non lo sarebbe.

Grazie al supporto e agli itemset è possibile definire delle regole di associazione volte a estrarre delle relazioni tra gli item all'interno di un dataset contenente più transazioni simili alla seguente:

pane $\rightarrow$ olio

che vuole indicare quante volte in una transazione che contiene l'item pane è presente anche l'item olio. Ogni regola di associazione è definita da uno o più antecedenti (nell'esempio mostrato l'item pane) e da uno o più conseguenti (in questo caso l'item olio). Ad ogni regola è associata una confidenza che è definita come il supporto dell'itemset contenente sia gli antecedenti che i conseguenti rispetto all'itemset contenente solo gli antecedenti. Supponendo di avere una regola generica con un solo antecedente (item A) e un solo conseguente (item B):

$$confidence = \frac{sup(A, B)}{sup(A)}$$

Nel caso relativo al dataset e alla regola presi in esame precedentemente:

$$confidence = \frac{sup(pane, olio)}{sup(pane)} = \frac{3}{4}$$

La regola presa in esempio ha dunque una confidenza di 0.75. Sia il supporto che la confidenza sono valori percentuali, riferiti a un numero più grande di transazioni rispetto a quello su cui sono calcolati e per questo variano tra un minimo di 0 e un massimo di 1.

L'estrazione di itemset e regole consiste quindi nell'estrarre delle associazioni all'interno delle transazioni di un dataset che soddisfino dei criteri di superamento di una soglia di supporto e confidenza minimo, dove per supporto della regola è inteso il supporto dell'itemset associato alla regola (contenente sia antecedenti che conseguenti) mentre per confidenza il rapporto visto precedentemente. Un algoritmo di estrazione delle regole si divide generalmente in due fasi: estrazione degli itemset (si possono usare algoritmi di tipo brute force, basati su livelli o sulla generazione di alberi) e estrazione delle regole di associazione (vengono calcolate tutte le partizioni binarie di ogni itemset frequente). Nel seguito di questo capitolo vedremo in dettaglio gli algoritmi più usati per l'estrazione degli itemset, uno dei quali è stato utilizzato nella realizzazione del lavoro di tesi.

2.2.1 Algoritmo Apriori

Escluso un approccio di tipo brute force (computazionalmente insostenibile), i due metodi principali utilizzati per l'estrazione degli itemset frequenti sono basati sull'algoritmo Apriori e sull'algoritmo FPGrowth. Il primo è basato su una struttura

a livelli e pone le sua fondamenta sul principio Apriori, il quale afferma che se un itemset è frequente allora anche tutti i suoi subsets sono anch'essi frequenti. Il meccanismo con cui questo algoritmo funziona, infatti, si basa sul considerare, a ogni livello, degli itemset composti da un numero sempre crescente di itemset, calcolandone il supporto e cancellando dai possibili itemset frequenti di quel livello quelli i cui subsets non sono contenuti negli itemset frequenti del livello precedente. Per spiegare meglio il funzionamento prendiamo ad esempio il dataset:

ID	item
1	pasta, olio
2	olio, sale, pepe
3	pane, sale, pepe, origano
4	pane, pepe, origano
5	pane, olio, sale
6	pane, olio, sale, pepe

Tabella 2.2. Esempio di dataset utilizzato per algoritmo Apriori

Al primo livello si considerano tutti gli itemset possibili contenenti al loro interno un item solo e si calcola il loro supporto:

itemset	sup
pane	5
olio	4
sale	4
pepe	4
origano	2

Tabella 2.3. itemset candidati di livello 1

A questo punto il supporto di ogni itemset candidato viene paragonato con la soglia limite scelta per considerare l'itemset frequente e, se maggiore, l'itemset corrispondente viene inserito nell'elenco degli itemset frequenti del livello considerato. Supponendo di scegliere come soglia 1, tutti gli itemset selezionati hanno supporto maggiore e quindi tutti vengono considerati come itemset frequenti di livello 1.

Il passo successivo consiste nel considerare tutti i possibili itemset composti da due item formabili utilizzando gli item di livello 1. Una volta generati i candidati anche in questo caso ne si calcola il supporto.

itemset	sup
pane, olio	3
pane, sale	3
pane, pepe	3
pane, origano	2
olio, sale	3
olio, pepe	2
olio, origano	0
sale, pepe	3
sale, origano	1
pepe, origano	2

Tabella 2.4. itemset candidati di livello 2

Siccome la soglia minima stabilita è pari a 1 i due itemset (olio, origano) e (sale, origano) non hanno un supporto sufficiente per essere considerati frequenti e vengono quindi scartati. L'elenco degli itemset frequenti di secondo livello è dunque:

itemset
pane, olio
pane, sale
pane, pepe
pane, origano
olio, sale
olio, pepe
sale, pepe
pepe, origano

Tabella 2.5. itemset frequenti di livello 2

Il passo successivo consiste nel calcolare gli itemset di livello 3, costituiti da tutte le combinazioni possibili formate da 3 elementi considerando gli itemset di secondo livello. Anche in questo caso, dopo aver calcolato quali sono gli itemset candidati, viene scannerizzato il dataset per calcolare il loro supporto e segnarlo in una tabella simile alle precedenti.

itemset	sup
pane, olio, sale	2
pane, olio, pepe	1
pane, olio, origano	0
pane, sale, pepe	2
pane, sale, origano	0
pane, pepe, origano	2
olio, sale, pepe	2
sale, pepe, origano	0

Tabella 2.6. itemset candidati di livello 3

In questo caso, oltre a non avere un supporto sufficiente (minore della soglia 1), gli itemset (pane, olio, origano), (pane, sale, origano) e (sale, pepe, origano) vengono scartati perché i loro subsets (olio, origano) e (sale, origano) non sono presenti negli itemset frequenti di livello 2. Anche se nella tabella mostrata è segnato il loro supporto, questi Itemests sono scartati dall'algoritmo ancora prima che esso venga calcolato, sfruttando il principio Apriori. L'itemset (pane, olio, pepe) viene invece scartato perché non ha un supporto maggiore o uguale alla soglia stabilita. L'elenco completo degli itemset frequenti di livello 3 risulta dunque:

itemset
pane, olio, sale
pane, sale, pepe
pane, pepe, origano
olio, sale, pepe

Tabella 2.7. itemset frequenti di livello 3

L'ultimo passo dell'algoritmo calcola gli itemset candidati di livello 4, esattamente come per i passi precedenti.

itemset	sup
pane, olio, sale, pepe	1

Tabella 2.8. itemset candidati di livello 4

Anche in questo caso il subset (pane, olio, pepe) non è presente negli itemset frequenti di livello 3 e quindi l'itemset non viene considerato come frequente; questo rende l'elenco degli itemset frequenti di livello 4 vuoto.

L'algoritmo Apriori funziona ed è molto utilizzato per analisi dati di questo tipo e estrazione di regole di associazione ma la sua efficienza, tuttavia, dipende dal numero di Item presenti nel dataset e, in particolare, dal tempo di generazione degli itemset candidati. Come descritto nell'esempio mostrato, inoltre, il dataset deve essere scannerizzato ogni volta che si passa a un livello successivo per contare il supporto degli itemset di quel livello. Tutte queste sottofasi richiedono un tempo che dipende dal dataset e non è predicibile; per questo motivo è stato pensato e introdotto negli engine più moderni e nelle librerie utilizzate per il data mining un algoritmo più efficiente chiamato FPGrowth.

2.2.2 Algoritmo FPGrowth

L'algoritmo FPGrowth è un algoritmo per la generazione di itemset frequenti che basa il suo funzionamento sulla costruzione di una struttura ad albero che modella il dataset considerato. In particolare, gli itemset frequenti vengono estratti percorrendo l'albero, senza dover quindi leggere il database più volte. Con l'algoritmo FPGrowth, a differenza di quello che avviene per l'algoritmo Apriori, il dataset viene scannerizzato solamente due volte: la prima per calcolare il supporto dei singoli item, la seconda per la costruzione dell'albero rappresentante il dataset e che prende il nome di FP-tree.

Per spiegarne il funzionamento prendiamo ad esempio il seguente dataset dove le transazioni sono composte da parole all'interno delle quali le lettere corrispondono ai singoli item:

TID	itemset
1	P, A, S, T, A
2	P, I, S, T, A
3	L, A, S, A, G, N, E
4	P, O, R, T, A

Tabella 2.9. Dataset di esempio per algoritmo FPGrowth

Il primo step dell'algoritmo FPGrowth è identico al primo step dell'algoritmo Apriori: si stabilisce una soglia minima di supporto, ogni item viene isolato e ne viene calcolato il supporto all'interno del dataset iniziale. Se il supporto dell'item è maggiore della soglia prefissata allora esso viene considerato dall'algoritmo, altrimenti viene scartato e non contribuisce alla costruzione dell'albero FP-tree. Nel caso in esempio il risultato di questo primo passaggio è il seguente:

Item	sup
P	3
A	4
S	3
T	3

Tabella 2.10. Item dopo primo step dell'algoritmo FPGrowth con relativo supporto

Da qui in avanti iniziano le differenze con l'algoritmo visto precedentemente. Per prima cosa gli item vengono ordinati in base al supporto secondo un ordine crescente:

Item	sup
T	3
S	3
P	3
A	4

Tabella 2.11. Item ordinati in base a supporto crescente

La stessa operazione viene effettuata sul dataset di partenza, considerando per ogni transazione solo gli item selezionati e ordinandoli in base al supporto secondo un ordine decrescente:

TID	itemset
1	A, P, S, T
2	A, P, S
3	A, S
4	A, P, T

Tabella 2.12. itemset ordinati per supporto decrescente degli item al loro interno

Una volta effettuata anche questa operazione inizia la costruzione vera e propria dell'albero. L'FP-tree è composto da una serie di nodi dove ogni nodo corrisponde a un item del dataset. Ogni transazione è registrata aggiungendo ogni item come figlio del nodo corrispondente all'item precedente, andando a costituire così un ramo dell'albero. Ogni item ha associato un contatore che stabilisce quante volte si

passa da quel nodo nella costruzione dell'albero. Prendendo per esempio la prima transazione filtrata con solo gli item con supporto maggiore alla soglia (A, P, S, T):

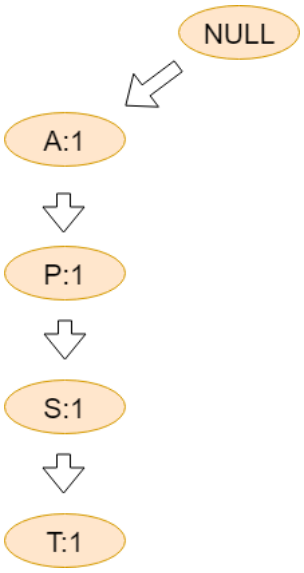

Figura 2.1. Costruzione dell'FP-tree dopo analisi della prima transazione del dataset

Dopo l'analisi della seconda transazione gli indici degli item vengono aumentati e l'albero diventa:

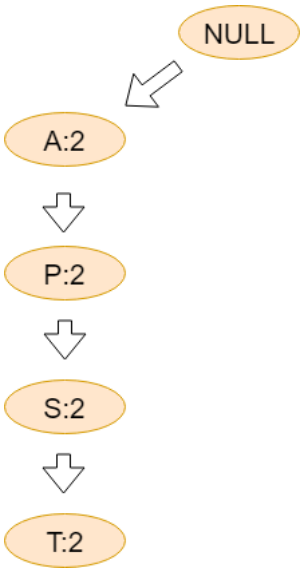

Figura 2.2. Costruzione dell'FP-tree dopo analisi della seconda transazione del dataset

Alla terza transazione, quella che considera l'itemset (A, S), il percorso effettuato

precedentemente non va più bene e viene inserito un nodo esterno dopo l'item A:

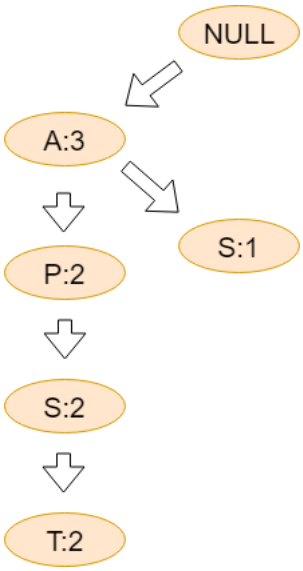

Figura 2.3. Costruzione dell'FP-tree dopo analisi della terza transazione del dataset

La stessa cosa vale anche per l'ultima transazione, dove dopo l'item P è presente subito l'item T:

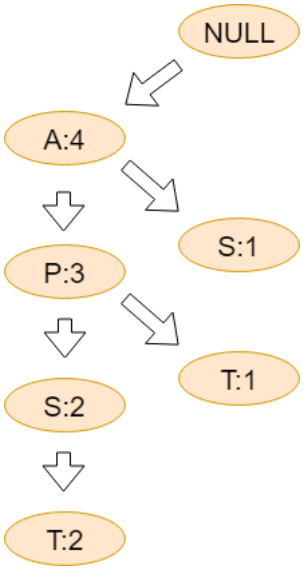

Figura 2.4. Costruzione dell'FP-tree dopo analisi della quarta transazione del dataset

Dopo che la costruzione dell'albero è conclusa, per ogni item escluso quello con il

supporto maggiore viene costruito il Conditional Pattern Base (CPB), ovvero l'elenco dei subset che lo precedono nell'albero. Il risultato finale è una tabella simile alla seguente:

Item	Conditional Pattern Base (CPB)
T	(APS : 1), (AP : 1)
S	(AP : 1), (A : 1)
P	(A : 3)
A	-

Tabella 2.13. Costruzione dei Conditional Pattern Base

Il passo successivo consiste nel selezionare, per ogni item, i subset comuni ai CPB determinati precedentemente e costruire quello che viene chiamato Conditional FP tree:

Item	Conditional Pattern Base (CPB)	Conditional FP tree
T	(APS : 1), (AP : 1)	(AP : 2)
S	(AP : 1), (A : 1)	(A : 2)
P	(A : 3)	(A : 3)
A	-	-

Tabella 2.14. Costruzione dei Conditional FP tree

Infine, la generazione degli itemset frequenti finali avviene combinando i Conditional FP tree trovati con l'item base a cui fanno riferimento, creando tutti i possibili subset:

Item	Conditional FP tree	Freq itemset
T	(AP : 2)	(AT : 2), (PT : 2), (APT : 2)
S	(A : 2)	(AS : 2)
P	(A : 3)	(AP : 3)
A	-	-

Tabella 2.15. Generazione degli itemset frequenti finali

L'algoritmo FPGrowth, come è possibile notare, è più efficiente rispetto all'algoritmo Apriori. Il tempo impiegato a costruire l'albero dipende dal numero di item e di transazioni presenti nel dataset iniziale, ma il numero di volte in cui il dataset viene scansionato è sempre pari a 2, qualsiasi sia il numero di record nel dataset; questo è il motivo per cui si è scelto di utilizzarlo nella realizzazione del lavoro di tesi.

2.3 Regole di associazione spazio-temporali

Come detto precedentemente, la quantità enorme di dati che ormai ci circonda ha attirato l'interesse del mondo del mercato e della ricerca. Questo ha portato a numerosi esperimenti, articoli e studi relativi alla gestione di big data. In questo contesto i dati spazio-temporali non fanno eccezione e l'argomento offre molti spunti e punti da cui partire per la creazione di algoritmi o sistemi per la loro analisi o visualizzazione. Questo tipo di dati, infatti, è presente in ogni aspetto immaginabile e, comprendendo una complessità di studio maggiore data dalla correlazione sia temporale che spaziale tra le informazioni, si è sempre alla ricerca di algoritmi più efficienti o adatti alla situazione che si vuole analizzare. In particolare la ricerca si orienta verso due filoni principali: lo studio e l'implementazione di algoritmi e sistemi sia per la gestione e il processamento dei dati che per l'estrazione di itemset o regole di associazione, e la realizzazione di applicazioni o linguaggi volti a favorire la visualizzazione di questo tipo di informazioni, rendendo la user experience sempre più semplice e intuitiva. Per il primo punto il limite ad oggi è prettamente concettuale più che tecnologico, la potenza di calcolo raggiunta ormai dagli elaboratori e dalle loro componenti o la possibilità di utilizzare cluster per le proprie analisi rende fattibile prettamente qualsiasi tipo di analisi sui dati. Rimangono aperte le sfide sull'efficienza di queste analisi, i tempi impiegati per analizzare i dati o la precisione ottenuta nel predire determinati eventi. Questo ha portato alla realizzazione di diversi contest e challenge dove sia studenti che professionisti si confrontano per proporre le loro soluzioni e migliorare lo stato attuale della ricerca. Per quanto riguarda la visualizzazione, invece, il trend sembra quello di realizzare o utilizzare programmi basati su tool per la gestione di informazioni spaziali standard come Google Earth o Google Maps, sia per le possibilità che il loro linguaggio proprietario offre (Keyhole Markup Language), sia per la facilità di utilizzo da parte degli utenti. L'alternativa principale a questi programmi è rappresentata dai linguaggi e ambienti di programmazione che offrono la possibilità di realizzare applicazioni grafiche, come ad esempio Java, Processing o WPF, purchè essi contengano delle librerie in grado di gestire mappe o informazioni spaziali.

Pur essendo un argomento molto studiato e i cui esperimenti sono iniziati da molti anni, sono ancora poche le definizioni e le formalizzazioni all'interno dell'ambito dei dati spazio-temporali. Tutte le analisi, tuttavia, si basano, come per i dati di tipo solo temporale, sulla ricerca di patterns e ricorrenze all'interno dei dataset considerati. In un articolo del 2018 Gowtham Atluri, Anuj Karpatne e Vipin Kumar stilano una lista di 5 possibili tipologie di patterns identificabili quando si analizzano dei dati di tipo spazio-temporale [1]:

1. Pattern di ricorrenza: identificati da subsets di eventi che accadono in prossimità spaziali e temporali vicine. Un esempio di dati analizzati dai quali è possibile estrarre patterns di questo tipo riguarda la relazione tra la chiusura dei bar in una città e le segnalazioni dovute alla guida in stato di ebbrezza. L'approccio più utilizzato per trovare dipendenze di questo tipo riguarda l'utilizzo di un algoritmo basato sul principio Apriori che calcola un coefficiente di co-occurrence spazio-temporale che funge da soglia per l'estrazione dei patterns.

2. Pattern sequenziali in punti spazio-temporali: patterns studiati in contesti nei quali sequenze di eventi spazio-temporali di una tipologia può generare eventi spazio-temporali di un'altra tipologia. Un esempio di pattern di questo tipo si ha nella relazione che intercorre tra un incidente stradale e la generazione di una coda nel traffico dopo poco tempo. Un approccio pensato per trovare patterns di questo tipo è stato studiato nel 2008 e porta all'estrazione di liste di eventi dove ogni elemento nella lista porta alla generazione del successivo.

3. Pattern sequenziali nelle traiettorie: sequenze di luoghi o regioni visitate da oggetti in movimento nello stesso ordine. Un esempio di questo tipo di pattern può essere visto dal percorso effettuato da turisti in una località visitata. Le traiettorie sono però oggetti particolari, infatti lo stesso percorso può essere rappresentato da strade che differiscono anche leggermente, non si avrà mai un'uguaglianza perfetta tra due percorsi. Per questo motivo, l'approccio più utilizzato per capire se due traiettorie sono la stessa è quello di dividere la zona geografica di interesse in macrozone e le singole traiettorie in segmenti. Se tutti i segmenti delle diverse traiettorie si trovano nelle stesse macrozone allora esse vengono considerate come lo stesso percorso. In questo caso è anche possibile inserire dei vincoli di tipo temporale per considerare come traiettorie simili solo quelle dove il tempo tra i vari luoghi visitati è lo stesso.

4. Pattern Motif: sono pattern che rappresentano ripetute misurazioni osservate a istanti temporali e luoghi diversi. I pattern motif possono essere individuati per informazioni anche solo temporali (eliminando le informazioni relative al luogo delle misurazioni) e in quel caso l'approccio per estrarre questo tipo di pattern consiste nella creaezione di una matrice di profilo calcolata tramite un algoritmo basato su una trasformata FFT (Fast Fourier Transform) e che determina delle sottosequenze temporali di cui viene calcolata la similarità. Nel caso di informazioni di tipo spazio-temporale l'algoritmo è modificato per tenere conto anche della correlazione spaziale tra gli eventi: lo stesso time-motif può presentarsi anche in luoghi vicini a quello dove è stato individuato. Un'analisi di questo tipo può essere fatta considerando per esempio la gestione di gruppi di ambulanze.

5. Pattern di rete: sono pattern estrapolati da dati spazio-temporali espressi tramite strutture a rete. Il modo principale di identificarli consiste nella creazione di communities nella rete dove ogni community rappresenta un gruppo di entità (nodi o link) che interagiscono tra loro più di quanto interagiscono con le altre entità

I pattern che il candidato ha cercato di estrarre durante il lavoro di tesi appartengono in particolare alla quarta categoria presentata, ma il modo in cui i dati sono stati analizzati e come questi sono stati ricavati verrà illustrato in seguito. Una delle difficoltà, ma anche uno dei vantaggi più importanti nell'analisi di dati di questo tipo riguarda la necessità di integrare tra loro dati di natura diversa. Sebbene infatti in tutti gli studi e esperimenti di data mining i dati sono spesso, per non dire quasi sempre, soggetti a una fase di preprocessing dove subiscono delle trasformazioni preliminari per essere preparati alle analisi, mai come per le informazioni spazio-temporali questo è vero e, soprattutto, necessario. L'integrazione che si effettua di solito riguarda sempre dati di tipo geografico e dati legati a un evento o al valore che si vuole analizzare, integrati in una struttura dati comune dove poi vengono effettuate le operazioni di estrazione di itemset e regole. Un esempio particolare di questa integrazione è avvenuta nel lavoro di Jeremy Mennis e Jun Wei Liu, due ricercatori americani che nel 2005 hanno provato a studiare la relazione tra il cambiamento geografico della zona di Denver e il suo legame con il cambiamento socio-economico avvenuto nella stessa zona nella seconda metà del XX secolo [2]. Per il loro lavoro sono stati utilizzati due dataset differenti: il primo derivante da un archivio di censimenti conservato nel tempo e il secondo invece ottenuto da un sistema di immagini satellitari chiamato GIS. I due sistemi sono stati integrati in una tabella a più livelli dove per ogni zona geografica erano associate le informazioni sul territorio e sul livello economico di appartenenza delle persone che la abitavano nel tempo. I due ricercatori hanno utilizzato, per separare tra loro le zone, un approccio molto simile a quello descritto nel punto relativo ai pattern sequenziali nelle traiettorie; l'intera area geografica analizzata, infatti, è stata divisa in macrozone rettangolari, ciascuna rappresentante una entry della tabella finale ottenuta dopo l'integrazione. Una volta ottenuto il dataset finale sono state estratte, tramite un algoritmo Apriori, le regole obiettivo dell'esperimento, che hanno rivelato delle correlazioni interessanti tra le aree a poca densità urbana e la percentuale di persone in stato di povertà, sottolineando anche il periodo dello sviluppo economico successivo agli anni '80 e l'impatto economico che la nascita dell'industria delle automobili ha avuto su una città come Denver. Ciò che manca, nell'esperimento dei due ricercatori Americani, è una visualizzazione dei risultati ottenuti; le regole, infatti, sono state analizzate tramite un file di testo ottenuto come output dell'esperimento, senza l'utilizzo di nessun tool grafico. Anche in questo senso, però, non sono mancati nel tempo degli esperimenti di visualizzazione e render grafico di itemset e regole di associazione legate a informazioni spaziali. Un esempio particolare è dato dal lavoro di un team di ricercatori italiani e irlandesi che, nel 2007, hanno sviluppato un sistema di più componenti il cui obiettivo era standardizzare il processo di analisi e visualizzazione di informazioni spazio-temporali, al cui interno è contenuto anche una parte atta e dedicata esclusivamente alla visualizzazione dei dati [3].

Il sistema proposto da questo esperimento si basa su tre componenti fondamentali: un data store al backend che contiene il dataset da analizzare e i dati in input, un application server centrale con un mining engine il cui scopo è processare le informazioni ed estrarre le regole e gli itemset richiesti e un blocco finale per la visualizzazione dei dati che può essere composto da tool e applicazioni grafiche di diversa natura.

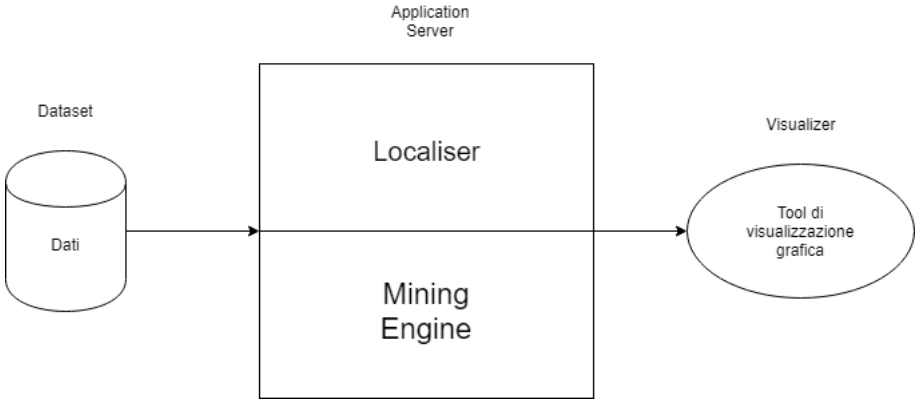

Figura 2.5. Schema componenti del sistema proposto in [3]

Come è possibile notare dalla figura, all'interno del blocco centrale del sistema, è presente, oltre al mining engine, anche un componente chiamato Localiser; l'idea alla base del sistema proposto nell'articolo, infatti, è quella di analizzare i dati spaziali e processarli per l'estrazione delle regole in modo parallelo, per diminuire il tempo di esecuzione e aumentare l'efficienza totale. Mentre il localiser preprocessa e prepara i dati, il mining engine li analizza e le due componenti lavorano in parallelo e contemporaneamente. I componenti sono stati testati su un caso studio relativo agli effetti che l'uragano Isabel ha avuto sulle coste del North Carolina nel 2003. La parte più importante dell'approccio riguarda tuttavia l'ultimo componente del sistema proposto e, in particolare, per il caso studio analizzato nell'esperimento, sono state pensate due modalità di visualizzazione delle informazioni e dei risultati estratti: il primo sfrutta Google Earth, il linguaggio KML e la possibilità di interagire con delle immagini ricostruite tramite dei satelliti; tramite delle query richieste dall'utente il sistema mostra all'utente i dati su una rappresentazione 3D della zona interessata, permettendo anche lo zoom e lo scrolling dell'immagine 3D renderizzata. Il secondo, invece, consiste in un programma realizzato appositamente per l'esperimento usando Java3D e serve all'utente per interagire con il mining engine e gestire la fase di estrazione delle regole, le quali sono poi visualizzate su una ricostruzione della zona a cui sono applicate o in formato XML.

Questi sono solo alcuni degli esempi di studi realizzati negli ultimi anni, sebbene ormai si siano fatti grandi passi in avanti in questo campo e molti strumenti siano stati realizzati (come ad esempio Apache Spark, un engine realizzato appositamente per l'analisi di big data), l'interesse per la ricerca sui dati spazio-temporali è ancora molto alto e sicuramente continuerà a esserlo per gli anni a venire.

2.3.1 Apache Spark

Apache Spark è un engine per il processamento di dati su larga scala sviluppato all'Università della California. Le caratteristiche principali a cui punta sono bassa

latenza, tolleranza ai fault. Spark viene utilizzato soprattutto per le analisi di dati che includono al loro interno dei processi iterativi, dove bisogna fare operazioni sugli stessi dati più volte.

Figura 2.6. Logo di Apache Spark

Per parlare dei vantaggi che Spark offre è necessario spiegare come funziona la principale alternativa ad esso, il framework MapReduce. Esso sfrutta Hadoop e, proprio come Spark, si appoggia a un file system distribuito denominato HDFS e a un cluster composto da più macchine. Nel caso di processi iterativi, MapReduce legge e scrive i dati sull'HDFS dopo ogni iterazione: questo permette un minore utilizzo della main memory ma coinvolge molte operazioni di I/O su disco che sono lente e che quindi rallentano il processo totale di analisi.

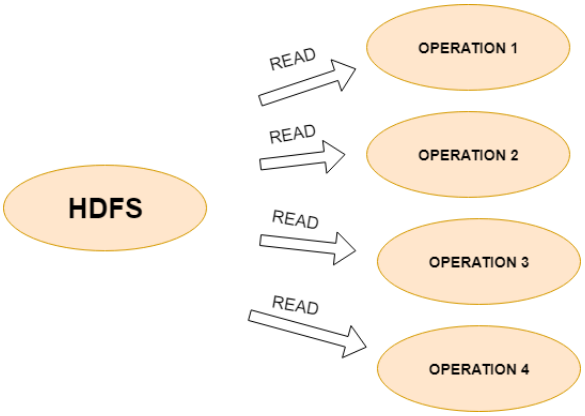

Figura 2.7. Schema funzionamento MapReduce

Il vantaggio principale e più grande di Spark riguarda il fatto che, rispetto all'algorirmo di MapReduce, sfrutta molto di più la main memory. In questo modo, diminuiscono le operazioni di I/O che devono essere fatte sull'HDFS e si ottengono prestazioni migliori, aumentando la velocità dell'algoritmo di anche 100 volte. Con Spark i dati sono presi una volta sola, all'inizio, dall'HDFS e salvati nella memoria principale. Successivamente vengono fatte su di essi tutte le operazioni necessarie per analizzarli. I dati nella main memory sono condivisi, parzialmente o totalmente, tra le iterazioni.

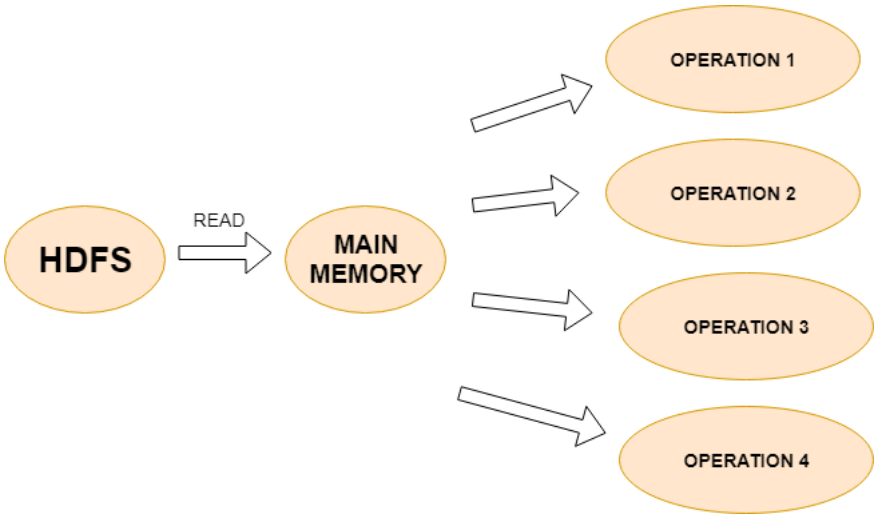

Figura 2.8. Schema funzionamento Spark

La rappresentazione dei dati in Spark avviene tramite dei Resilient Distributed Datasets che possono essere descritti come collezioni di oggetti partizionate e distribuite condivise dai nodi di un cluster. Su questi RDDs possono essere operate delle trasformazioni o delle azioni per manipolare i dati. I programmi scritti con Spark consistono in del codice che viene eseguito in parallelo sugli RDDs e che compie delle trasformazioni o delle operazioni su di essi. Architetturalmente Spark si basa su un componente di basso livello chiamato Spark Core che esplita le funzioni di gestione della memoria, schedulazione dei task e recupero dei fault. Su di esso si istanziano dei componenti di più alto livello che permettono l'interazione con i dati e la loro analisi:

1. Spark SQL: permette l'interazione con diversi formati di dato tramite il linguaggio SQL e HQL. Spark può interagire con file csv, JSON, tabelle di database andando a creare delle strutture dati mappate da alcune classi contenute in questo componente.

2. Spark Streaming: componente utilizzato per l'analisi di dati live, permette di specificare una finestra temporale all'interno della quale poter fare delle misurazioni o analisi su dei dati che arrivano in tempo reale da un server.

3. GraphX: una libreria per il processamento e la manipolazione dei grafi.

4. MLLib: una libreria per il Machine Learning e il Data Mining, pensata per implementare la versione parallela di algoritmi di clustering, classificazione, estrazione di itemset e regole di associazione.

Il lavoro di tesi proposto e l'applicazione sviluppata per l'analisi dei dati fanno largo uso, in particolare, dell'ultimo componente descritto.

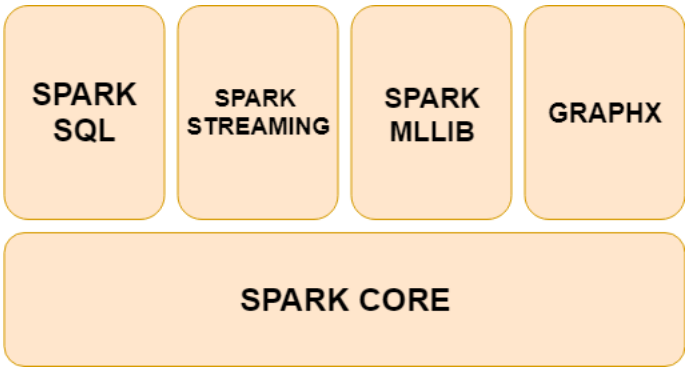

Figura 2.9. Schema architetturale Spark

Spark prevede e implementa la possibilità di utilizzare diversi linguaggi di programmazione, come ad esempio Java, Python e Scala. Per il lavoro presentato è stato scelto il linguaggio Java, sia per conoscenze pregresse del candidato che per la maggiore flessibilità e comodità rispetto agli altri linguaggi.

Capitolo 3

Analisi di eventi georeferenziati

3.1 Descrizione del problema

Il lavoro di sviluppo della tesi presentata è stato incentrato sulla realizzazione di un algoritmo per l'estrazione di regole di associazione spaziali-temporali relative allo stato di punti geograficamente localizzati in diversi istanti temporali. Più precisamente, l'obiettivo è stato quello di capire come estrapolare da un dataset iniziale delle regole che permettessero di mettere in relazione tra loro eventi che avvengono in diversi luoghi e in diversi istanti temporali. Un esempio di evento può essere, per esempio e come considerato nel caso studio trattato in questa tesi, lo stato di alcune stazioni per biciclette:

1. FULL (se il numero di posti liberi è minore di una certa soglia),
2. EMPTY (se il numero di posti occupati è minore di una certa soglia)
3. NORMAL (se nessuna delle due condizioni precedenti è verificata).

Più generalmente l'analisi di qualsiasi tipo di evento, per ragioni logistiche, viene effettuata su una finestra temporale lunga un numero finito di istanti. Supponendo di assegnare il numero 0 al al delta temporale dal primo evento della sequenza generata, l'obiettivo è quello di studiare il comportamento di altri luoghi legati all'evento negli istanti successivi, anch'essi identificati da un numero. Per descrivere univocamente lo stato di un determinato luogo all'interno della finestra temporale è dunque necessario sapere tre cose:

1. L' istante temporale considerato (partendo da 0)
2. L' identificatore del luogo dove si verifica l'evento
3. Lo stato dell'evento considerato

È possibile comprimere questi tre parametri per descrivere lo stato di un luogo con la sintassi:

0 1 event

la quale indica che nel luogo con indice 1 all'istante temporale 0 si verifica un evento di tipo event.

Supponendo di avere quindi tre luoghi identificati dagli indici 1, 2 e 3 quello che si è cercato di ottenere è stato una serie di regole simili alla seguente:

0 1 event1, 1 2 event2 -> 2 3 event3

che potrebbe essere tradotta con l' espressione verbale: "Se nell luogo 1 si verifica l'evento event1 e nel luogo 2 all'istante temporale di riferimento successivo si verifica l'evento event2, allora nel luogo 3 dopo 2 istanti temporali si verificherà l'evento event3". Supponendo per esempio di avere i seguenti record:

10:10_27/05/2006 p1 event1
10:15_27/05/2006 p2 event2
10:20_27/05/2006 p3 event3
10:25_27/05/2006 p2 event6
10:30_27/05/2006 p1 event1
10:35_27/05/2006 p2 event2
10:40_27/05/2006 p3 event3

le sequenze temporali lunghe 3 istanti che partono alle 10:10 e alle 10:30 generano lo stesso itemset che può essere trasformato nella regola:

0_p1_event1, 1_p2_event2 -> 2_p3_event3

3.2 Strumenti e tecnologie utilizzate

Lo sviluppo della tesi presentata ha previsto l'interazione e la cooperazione di diverse tecnologie e linguaggi di programmazione. Esistono molti strumenti per l'analisi di big data e per il data mining in generale, si possono nominare per esempio Hadoop, RapidMiner, così come esistono diversi linguaggi di programmazione sia per l'analisi di dati (Python, R, Scala) che per la visualizzazione di informazioni grafiche. Le motivazioni della scelta di alcune tecnologie rispetto ad altre sono state prese sia per una conoscenza pregressa di alcune componenti (e quindi un maggior risparmio di tempo), sia per il tipo di analisi e misurazioni che dovevano essere svolte durante il lavoro, il che ha comportato una fase preliminare di studio che ha occupato la prima parte del periodo di tesi. Verranno di seguito elencati i principali strumenti che sono stati utilizzati durante il periodo di lavoro, con annessa descrizione.

3.2.1 Linguaggio Java

Java è un linguaggio di programmazione orientato agli oggetti e basato su classi creato da un piccolo gruppo di ignegneri dell'azienda Sun nel 1991 guidati da James Gosling. Esso si basa sul concetto che il codice scritto con questo linguaggio possa essere eseguito su qualsiasi macchina indipendemente dal processore che essa possiede. Per realizzare ciò il codice scritto in Java è compilato in un linguaggio intermedio chiamato Java Bytecode che viene poi converito da una macchina virtuale detta Java Virtual Machine (JVM) della macchina dove si vuole far funzionare il programma nel codice macchina adatto al processore.

Figura 3.1. Logo Java

La sintassi del linguaggio Java deriva in gran parte dal C e dal C++, ma possiede alcune facilitazioni come la gestione automatica della memoria e il rilascio delle risorse (grazie a un oggetto detto Garbage Collector), cosa che invece negli altri linguaggi deve essere implementata dal programmatore. Java offre inoltre il supporto al multithreading, ovvero la possibilità di creare dei programmi in cui alcune operazioni vengono svolte in parallelo, senza che l'applicazione resti bloccata nell'attesa del risultato di un'operazione e rendendola così più veloce ed efficiente.

Come detto precedentemente il linguaggio Java è un linguaggio orientato agli oggetti, in particolare si possono identificare due tipi di dati:

1. Tipi primitivi: sono i tipi di dato base utilizzati anche in altri linguaggi non orientati agli oggett. Un esempio sono int, float, double.

2. Tipi riferimento: definiscono degli oggetti e in Java sono implementati tramite classi e interfacce. Le classi contengono uno stato e dei metodi (posseggono sia delle variabili che delle funzioni) mentre le interfacce posseggono solo dei metodi (vengono estese dalle classi in modo che vi siano in esse già alcuni metodi e che il programmatore debba scriverne solo l'implementazione e non la definizione)

L'orientamento agli oggetti permette inoltre che il linguaggio supporti l'ereditarietà tra le classi. Con questo termine si intende la possibilità da parte di un oggetto di estendere le caratteristiche di un altro. A differenza però di linguaggi come il C++ si tratta di un tipo di ereditarietà singola, una classe può implementare tutte le interfacce che si desiderano ma estendere solamente un'altra classe.

Una delle conseguenze dell'ereditarietà è il cosiddetto Poliformismo. Con questo termine si indica la possibilità di ridefinire, all'interno di una classe che ne estende un'altra, alcuni dei metodi della classe estesa tramite un'operazione che prende il nome di Override.

Per poter scrivere ed eseguire un programma Java servono principalmente due cose: il Java Runtime Environment (JRE) e il Java Development Kit (JDK) che nelle ultime versioni comprende al suo interno il JRE. Il primo offre un'implementazione della piattaforma Java utile all'esecuzione dei programmi, mentre il secondo serve a scrivere i programmi e contiene al suo interno una serie di tool utili agli sviluppatori, tra cui Javadoc. Una delle componenti fondamentali di un linguaggio è la possibilità di poter creare della documentazione atta a spiegare a chi legge il codice o chi deve programmarlo cosa fanno determinate funzioni: Javadoc è un tool per Java che permette, tramite delle annotazioni, di trasformare ciò che viene scritto dal programmatore nel codice come commento in della vera e propria documentazione.

I motivi per cui si è scelto di usare Java per questa tesi sono diversi. Innanzitutto per conoscenza pregressa del candidato, in secondo luogo per le caratteristiche e l'interazione che Java offre, in particolare con Spark. Esistono molte librerie di terze parti scritte in Java ed essendo un linguaggio molto utilizzato si può considerare uno standard in diversi ambiti. Il lavoro presentato fa utilizzo di alcune di queste librerie, in particolare di una libreria open source sviluppata per essere compatibile con Java e realizzare applicazioni grafiche.

Come per tutti i linguaggi, per poter scrivere un programma in Java è necessario, oltre al JDK e al JRE, un ambiente di sviluppo integrato (IDE). Ne esistono molti in commercio, sia proprietari che gratuiti e i più utilizzati sono IntelliJ IDEA, NetBeans (sponsorizzato da Oracle e open source) ed Eclipse, che è stato utilizzato dal candidato per lo sviluppo della tesi proposta insieme alla versione 8 di Java (l'ultima release ufficiale è la Java SE 11).

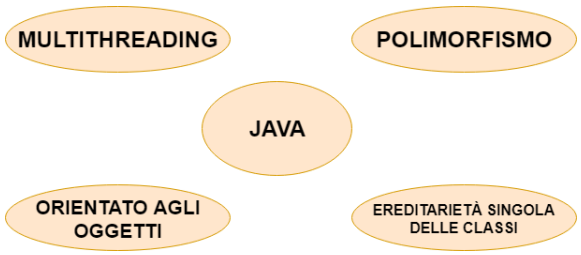

Figura 3.2. Principali caratteristiche Java descritte

3.2.2 JavaFX

JavaFX è una libreria compatibile con Java pensata e creata per lo sviluppo di applicazioni grafiche. Fino alla versione 8 di Java era possibile creare delle applicazioni grafiche tramite la libreria ufficiale Swing, composta da oggetti e classi scritti direttamente in codice Java e che permetteva la creazione di applicazioni grafiche tramite

la composizione di una finestra mediante widget e elementi base (per esempio bottoni e label). Swing si basa sul modello MVC (Model View Controller), secondo cui ogni aspetto dell'applicazione è modellato e descritto da tre classi o componenti fondamentali:

1. Model: si occupa dell'interazione dell'applicazione, modellandone le operazioni fondamentali di creazione, cancellazione, modifica e lettura (chiamate anche operazioni CRUD)

2. View: gestisce tutto ciò che riguarda il comportamento grafico dell'aspetto modellato, i componenti visibili all'utente e gestisce l'interazione tra esso e il componente descritto.

3. Controller: mette in comunicazione tra loro la View e il Model, implementando i metodi che vengono chiamati dalla prima e che servono a operare delle operazioni sui dati tramite il secondo. Può implementare dei filtri, controllare che l'utente possa fare l'operazione per cui ha richiesto l'interazione svolgendo dunque una fase di controllo.

In Swing ogni componente grafico ha associate una classe che ne rappresenta il modello e una che ne rappresenta il controller, e l'interazione delle 3 viene specificata dal programmatore. Questa libreria, tuttavia, non si è rivelata molto intuitiva da parte degli utenti e la realizzazione di applicazioni grafiche risultava complessa anche per operazioni molto semplici, per esempio vi era bisogno di molte righe di codice per descrivere il comportamento dell'applicazione alla pressione di un bottone.

Da questa esigenza nasce la libreria JavaFX, che si basa anch'essa sul modello MVC, ma è molto più semplice da usare per il programmatore. La libreria JavaFX è composta da una serie di classi che rivedono i principali componenti che erano presenti anche in Swing facilitandone l'utilizzo anche grazie alle Lambda Functions (espressioni introdotte in Java 8 che permettono di specificare una funzione che deve essere eseguita una volta solo in una forma compatta all'interno di un metodo senza doverne scrivere la signature). In questo modo, se si considera l'esempio del bottone fatto precedentemente, JavaFX metta a disposizione una funzione che viene chiamata quando l'utente genera un'azione sul bottone e il programmatore può semplicimente specificare, tramite una Lambda Function, al suo interno il comportamento che dovrà avere l'applicazione. La novità principale però di JavaFX è l'introduzione di un linguaggio di markup chiamato FXML, tramite il quale è possibile legare più facilmente tra loro la parte grafica dell'applicazione e la parte logica che gestisce l'applicazione, in un modo simile a come avviene per la programmazione web tramite il binding dei dati. Questo permette di specificare in un file FXML tutte le componenti grafiche dell'applicazione e legarle ad un unico controller che gestisce i metodi e le chiamate alle funzioni della parte logica. Se in Swing per ogni componente si aveva una definizione di modello, view e controller, un'applicazione JavaFX può essere vista come definita da tre macroblocchi:

1. Main: la classe principale che si occupa di fare lo startup dell'applicazione e di generare il Controller

2. FXController: la classe che gestisce la logica dell'applicazione e al cui interno si ha un riferimento agli elementi grafici presenti nel file FXML

3. FXML: il file che contiene le informazioni grafiche relative all'aspetto visivo dell'applicazione, con i riferimenti alle chiamate del controller.

È possibile anche aggiungere un quarto elemento relativo allo styling dell'applicazione. Proprio come avviene nelle applicazioni web si può inserire all'interno del file FXML il codice relativo al CSS che ogni componente grafico possiede, oppure un link a un file esterno dove sono definite le principali classi di stile che si vogliono utilizzare nell'applicazione.

JavaFX si basa su componenti grafici e, per questo motivo, è possibile utilizzare quelli previsti nella libreria oppure crearne di nuovi e custom, così come è possibile creare delle vere e proprie librerie compatibili. Per il lavoro di tesi presentato, in particolare, è stata utilizzata una libreria open source chiamata GMapsFX, sviluppata per funzionare come Wrapper alle API Javascript di Google Maps. L'ultima release rilasciata è la 2.12.0, ma per l'applicazione realizzata durante il lavoro di tesi si è utilizzata la versione 2.10.0. Questa libreria permette di interagire con una mappa di Google Maps e di poter svolgere su di essa alcune operazioni fondamentali. Il Wrapper crea un component Map compatibile con JavaFX che può quindi essere chiamato dal programmatore come un qualsiasi altro componente grafico. La libreria mette a disposizione delle funzioni per operare con questo componente, in particolare permette di settare il punto iniziale di visualizzazione della mappa o di inserire dei marker per contraddistinguere dei luoghi su di essa, cosa che è stata ampiamente utilizzata nel corso del lavoro presentato.

Figura 3.3. Esempio del componente Map messo a disposizione dalla libreria GMapsFX

3.2.3 Linguaggio KML

L'ultimo strumento coinvolto nella realizzazione del lavoro di tesi è stato il linguaggio KML, utilizzato per una parte della visualizzazione dei risultati in alternativa

all'applicazione sviluppata con JavaFX. Il KML (Keyhole Markup Language) è un linguaggio sviluppato appositamente per la visualizzazione grafica di dati spaziali su mappe 2D e 3D. Essendo un linguaggio di markup il suo principale utilizzo è nell'ambito della programmazione web e infatti è utilizzato come standard dai principali tool di visualizzazione di informazioni spaziali online, quali Google Earth e Google Maps. Un file KML può essere creato con un normale editor di testo e, tramite il browser, caricato attraverso delle apposite sezioni sui siti dei tool descritti sopra che lo interpretano e permettono all'utente, in poco tempo, di visualizzare le informazioni desiderate. Tramite il linguaggio KML è possibile fare delle operazioni più o meno complesse sulle mappe messe a disposizione. In particolare, è possibile creare dei poligoni che definiscono una zona interessata, creare dei percorsi o posizionare dei marker su dei luoghi designati. Il candidato ha fatto utilizzo particolare di questo ultimo punto, come verrà spiegato in seguito. Un file KML può anche interagire con del codice Javascript per creare delle vere e proprie applicazioni web interattive; allo stesso modo è possibile stilizzare tutti i componenti che si inseriscono nella mappa, come ad esempio il colore dei placemarker o dei percorsi. Questo avviene inserendo all'interno dei tag che definiscono i componenti del codice inline oppure linkando il file a un foglio di stile CSS.

Figura 3.4. Risultato di un file KML interpretato e caricato su Google Earth

```
<kml xmlns="http://www.opengis.net/kml/2.2"><Document>
<Placemark>
  <name>44</name>
  <ExtendedData>
    <Data name="DayWeek">
      <value>Mon</value>
    </Data>
    <Data name="Hour">
      <value>3</value>
    </Data>
    <Data name="Criticality">
      <value>0.5440729483282675</value>
    </Data>
  </ExtendedData>
  <Point>
      <coordinates>2.189700,41.379047</coordinates>
  </Point>
</Placemark>
</Document></kml>
```

Figura 3.5. Esempio di un semplice file KML per la visualizzazione di un placemarker

3.3 Preprocessing dei dati

Il lavoro illustrato nel documento ha previsto lo sviluppo di tre blocchi principali: il primo relativo al preprocessing dei dati, il secondo atto all'estrazione delle regole e l'ultimo, invece, pensato per la visualizzazione dei risultati.

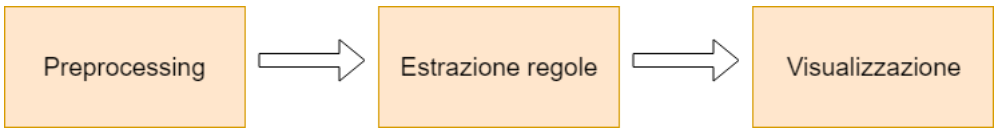

Figura 3.6. Schema macroblocchi realizzati

Come detto precedentemente, spesso i dati ricevuti in input quando si lavora nell'ambito del data mining possono avere delle impurità o non essere nel formato più consono o utile per il lavoro da svolgere. Per questo motivo, prima di iniziare a sviluppare un algoritmo per la loro analisi, è necessario attraversare una fase di preprocessing dei dati, dove essi vengono ripuliti o trasformati nel formato più adatto per le elaborazioni che si devono svolgere.

Supponendo che nelle misurazioni di tipo spazio-temporale le informazioni necessarie e che sono presenti in ogni dataset da analizzare siano: data e ora della misurazione, id associato al luogo della misurazione ed evento associato alla misurazione, si possono applicare delle trasformazioni per trasformare eventuali dati in più nell'evento specifico che si vuole identificare. Per la tesi proposta, ad esempio, per prima cosa si è cercato di eliminare l'informazione relativa al numero di posti liberi o occupati in una stazione e ottenere invece un'informazione sullo stato di quest'ultima. Specificata infatti una soglia dall'utente, il programma analizza i dati in input verificando se il numero dei posti liberi o occupati è inferiore alla soglia e determinando quindi in che stato si trova la stazione in quell'istante. Prendendo per esempio il record:

1 2008-05-15 12:01:00 0 18

e specificando come soglia 3, poiché il numero di posti liberi è 0 il risultato ottenuto sarà:

1 2008-05-15 12:01:00 full

Per rendere l'estrazione delle regole più semplice si può scegliere, nel caso le misurazioni siano state fatte con sensibilità ai secondi, di approssimare l'orario della misurazione al minuto intero pari inferiore più vicino, senza tenere conto dell'informazione sui secondi. Nel caso dell'esempio precedente il risultato dopo la trasformazione sarà il seguente:

1 2008-05-15 12:00 full

L'algoritmo proposto per la tesi è stato pensato per essere utilizzato con qualsiasi tipo di dato in formato testuale. Per questo motivo si è scelto di inserire nella fase di pre-processing dei dati una trasformazione volta a standardizzare il formato dei dati ricevuti in input, in modo che successivamente l'algoritmo di estrazione delle regole possa lavorare su qualsiasi tipo di dati che sia stato trasformato in quel formato. I record ottenuti devono avere una struttura simile:

hh:mm_YYYY/MM/DD id event

dove i vari campi rappresentano:

1. hh: ora della misurazione
2. mm: minuto della misurazione
3. DD: giorno della misurazione
4. MM: mese della misurazione
5. YYYY: anno della misurazione
6. id: identificativo del luogo della misurazione
7. event: evento associato alla misurazione

La trasformazione successiva, dunque, è volta proprio a trasformare il record di esempio nel formato descritto, ottenendo come risultato:

12:00_2008/05/15 1 event

L'ultima fase di pre-processing è quella che descrive l'idea principale della tesi, ovvero come trasformare dei record relativi a una singola stazione in dei record che contengano le informazioni su un insieme di stazioni in una finestra temporale specificata. Per fare questo si è utilizzato Spark e la possibilità che esso mette a

disposizione di generare, partendo da un record, una o più coppie chiave - valore. Il programma analizza ogni record dei dati ottenuti dalla trasformazione precedente e, per ognuno di essi, genera un numero di coppie chiave-valore pari al valore della finestra temporale specificata dall'utente (valore che viene passato come parametro al programma). Ciascuna di queste coppie ha come chiave l'orario della misurazione concatenato alla data e come valore un record nel formato:

istanteTemporale idLuogo evento

Lo schema seguente mostra un esempio della trasformazione nel caso di una finestra temporale pari a 3 per il caso studio relativo alle stazioni di biciclette:

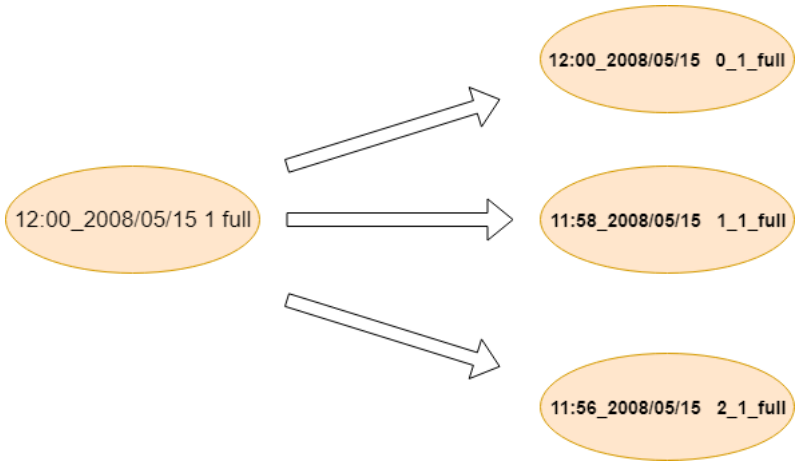

Figura 3.7. Generazione delle tuple con finestra temporale 3.

Il programma tiene conto della finestra temporale e genera per ogni record, ricorsivamente, una coppia chiave-valore dove la chiave contiene l'orario di misurazione precedente all'ultimo generato e il valore un numero che specifica qual è il delta temporale tra il momento in cui si verifica l'evento della misurazione e la data precedente generata.

Prendendo per esempio il record:

12:00_2008/05/15 1 full

Le prime due coppie generate saranno:

(12:00_2008/05/15, 0_1_full)
(11:58_2008/05/15, 1_1_full)

```
declare list of tuples
declare list of dates

for every temporal istant in the window

     if first temporal instant

            add to list of tuples a tuple with key=minutes_date and value = temporalIstant_idLocation_event
            add to list of dates minutes_date

     else

            calculate previous date
            add to list of tuples a tuple with key=minutes_date and value = temporalIstant_idLocation_event
            add to list of dates pevious date
```

Figura 3.8. Pseudocodice generazione delle tuple

La seconda coppia ha, nel valore, istante temporale 1 anziché 0 poiché tra il record delle 11:58 e quello delle 12:00 c'è un istante temporale di differenza. Seguendo questo ragionamento e avendo una finestra temporale pari a 3 l'ultima coppia generata sarà:

(11:56_2008/05/15, 2_1_full)

perché tra la misurazione delle 11:56 e quella delle 12:00 ci sono due istanti temporali di differenza.

Una volta che tutti i record sono stati trasformati in questo modo il programma li raggruppa in base alle chiavi, in modo da ottenere per ogni istante di misurazione l'elenco delle triplette istanteTemporale_idLuogo_event nella finestra temporale. Il risultato finale (considerando solo i valori delle coppie) è simile al seguente:

```
0_7_full 1_7_full 2_7_full 0_8_full 1_8_full 2_8_full 0_9_normal 1_9_normal 2_9_normal 0_10_full 1_10
_full 2_10_full 0_11_full 1_11_full 2_11_full 0_12_normal 1_12_normal 2_12_normal 0_1_normal 1_1_norm
al 2_1_normal 0_2_full 1_2_full 2_2_full 0_3_normal 1_3_normal 2_3_normal 0_4_normal 1_4_normal 2_4_n
ormal 0_5_full 1_5_full 2_5_full 0_6_normal 1_6_normal 2_6_normal
0_7_empty 1_7_empty 2_7_empty 0_8_empty 1_8_empty 2_8_empty 0_9_empty 1_9_empty 2_9_empty 0_10_empty
 1_10_empty 2_10_empty 0_11_empty 1_11_empty 2_11_empty 0_12_empty 1_12_empty 2_12_empty 0_1_full 1_1
_full 2_1_full 0_2_normal 1_2_normal 2_2_normal 0_3_empty 1_3_empty 2_3_empty 0_4_empty 1_4_empty 2_4
_empty 0_5_normal 1_5_normal 2_5_normal 0_6_empty 1_6_empty 2_6_empty
0_7_empty 1_7_normal 2_7_normal 0_8_normal 1_8_normal 2_8_normal 0_9_empty 1_9_empty 2_9_empty 0_10_e
mpty 1_10_empty 2_10_empty 0_11_normal 1_11_normal 2_11_normal 0_12_normal 1_12_normal 2_12_normal 0_
1_empty 1_1_empty 2_1_empty 0_2_empty 1_2_empty 2_2_empty 0_3_empty 1_3_empty 2_3_empty 0_4_empty 1_4
_empty 2_4_empty 0_5_empty 1_5_empty 2_5_empty 0_6_empty 1_6_empty 2_6_empty
0_7_full 1_7_full 2_7_normal 0_8_normal 1_8_normal 2_8_normal 0_9_empty 1_9_empty 2_9_empty 0_10_empt
y 1_10_empty 2_10_empty 0_11_normal 1_11_normal 2_11_normal 0_12_normal 1_12_normal 2_12_normal 0_1_n
ormal 1_1_normal 2_1_normal 0_2_normal 1_2_normal 2_2_normal 0_3_empty 1_3_empty 2_3_empty 0_4_empty
```

Figura 3.9. Elenco degli stati delle stazioni per ogni istante temporale.

3.4 Processing dei dati

Una volta effettuate tutte le trasformazioni preliminari descritte nella sezione precedente il programma inizia la vera e propria fase di processamento dei dati per l'estrazione delle regole. Apache Spark mette a disposizione una libreria chiamata Spark MLLib che contiene al suo interno due funzioni principali che permettono di estrarre gli itemset frequenti e le regole di associazione, specificando come parametri il supporto e la confidenza minima degli itemset e delle regole che si vogliono ottenere. Gran parte del lavoro di sperimentazione è consistito nello studiare il comportamento dell'algoritmo al variare di questi parametri. Lo schema seguente mostra i passi principali della fase di processing dei dati:

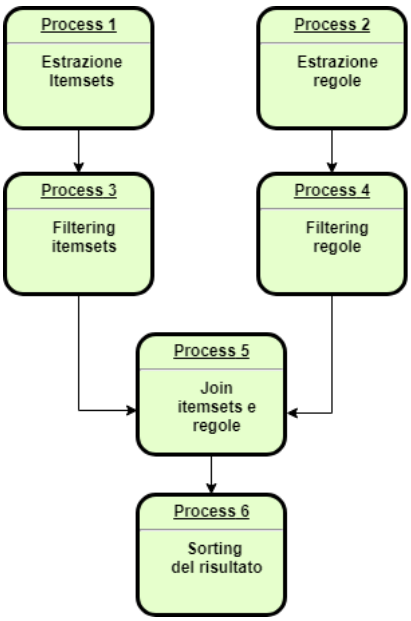

Figura 3.10. Principali passi della fase di processing dei dati.

Nel prosieguo di questa sezione ogni passo verrà analizzato singolarmente, in modo da poter comprenderne il funzionamento e perché si è scelto di inserirlo.

3.4.1 Estrazione degli itemset

La prima operazione che viene effettuata sui dati ottenuti dopo la fase di preprocessing è quella di dichiarare qual è il supporto e la confidenza minima che si vogliono utilizzare per l'estrazione di itemset e regole. Questo viene fatto utilizzando la classe FPGrowth della libreria MLLib di Spark, che permette di creare un oggetto che modella un algoritmo di Frequent Pattern Mining per l'estrazione degli itemset frequenti e delle regole. Il supporto che si può specificare alla funzione varia tra un

valore minimo di 0 e un valore massimo di 1 e rappresenta dunque un supporto percentuale. Scegliendo per esempio come valore 0.2 verranno considerati come itemset frequenti solo i sottoinsiemi dei record che appaiono in un numero di record pari ad almeno il 20% dei record totali. Il risultato è un file di testo con un aspetto simile al seguente:

```
2_5_empty, 2_10_full, 2_11_full, 1_11_full, 0_11_full , 9819
2_5_empty, 2_10_full, 2_11_full, 0_11_full , 9872
2_5_empty, 2_10_full, 2_9_full, 2_11_full, 1_11_full, 0_11_full , 9361
2_5_empty, 2_10_full, 2_9_full, 2_11_full, 0_11_full , 9408
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 2_11_full, 1_11_full, 0_11_full , 9317
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 2_11_full, 0_11_full , 9343
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 1_11_full, 0_11_full , 9391
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full , 10909
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 2_11_full , 9428
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 2_11_full, 1_11_full , 9342
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 2_11_full, 1_11_full, 0_11_full , 9276
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 2_11_full, 0_11_full , 9302
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 1_11_full , 9433
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 1_11_full, 0_11_full , 9348
2_5_empty, 2_10_full, 2_9_full, 1_9_full, 0_9_full, 0_11_full , 9438
```

Figura 3.11. Esempio di file ottenuto dopo l'estrazione degli itemset.

Il file contiene, per ogni riga, l'elenco degli stati delle stazioni e, a destra, un numero che indica il supporto assoluto di quel sottoinsieme di record rispetto al numero di record totali considerati inizialmente. Gli stati della prima riga, per esempio, compaiono 9819 volte in totale nei dati considerati prima dell'estrazione degli itemset frequenti. In questa fase di estrazione iniziale gli stati presenti nel sottoinsieme non sono ordinati in alcun modo, né in base al supporto né in base all'istante temporale a cui si riferiscono, cosa che è invece stata fatta in una fase successiva. Il supporto, che in questa fase è un supporto assoluto, sarà poi trasformato in seguito, durante la fase di visualizzazione, in un supporto relativo andando a dividere il suo valore per il numero di record totali.

Il risultato mostrato in figura è stato preceduto, dopo l'estrazione degli itemset, da una trasformazione preliminare volta a rendere più leggibili i record nel file (per esempio inserendo degli spazi e delle virgole tra uno stato e l'altro o eliminando alcune parentesi che Spark inserisce di default dopo l'estrazione).

3.4.2 Estrazione delle regole

Come detto precedentemente, l'oggetto FPGrowth creato per il mining dei dati permette anche di specificare la confidenza minima delle regole che si vogliono estrarre. Una volta inserito il valore, il programma, dopo aver estratto gli itemset frequenti, inizia l'estrazione delle regole. Anche in questo caso il valore da inserire varia tra un minimo di 0 e un massimo di 1, rappresentando dunque una confidenza percentuale.

```
(0_6_empty, 1_3_empty, 1_4_empty) , 2_6_empty , 0.9610370035420905
(0_6_empty, 1_3_empty, 1_4_empty) , 2_5_empty , 0.8211621071670812
(0_6_empty, 1_3_empty, 1_4_empty) , 2_3_empty , 0.983495365136785
(0_6_empty, 1_3_empty, 1_4_empty) , 2_1_empty , 0.6955309367699148
(0_6_empty, 1_3_empty, 1_4_empty) , 2_4_empty , 0.9818373652875122
(0_6_empty, 1_3_empty, 1_4_empty) , 2_11_full , 0.688748210113799
(0_6_empty, 0_5_empty, 1_4_empty) , 2_6_empty , 0.9712877030162413
(0_6_empty, 0_5_empty, 1_4_empty) , 2_5_empty , 0.9695475638051044
(0_6_empty, 0_5_empty, 1_4_empty) , 2_3_empty , 0.7925609048723898
(0_6_empty, 0_5_empty, 1_4_empty) , 2_4_empty , 0.978393271461717
(0_6_empty, 0_5_empty, 1_4_empty) , 2_1_empty , 0.7045388631090487
(0_6_empty, 0_5_empty, 1_4_empty) , 2_10_full , 0.6645881670533643
(0_6_empty, 0_5_empty, 1_4_empty) , 2_11_full , 0.7213602088167054
```

Figura 3.12. Esempio di file ottenuto dopo l'estrazione deglle regole.

Spark permette di visualizzare gli antecedenti e i conseguenti delle regole separandoli mediante delle parentesi tonde. Per ogni regola è specificata anche la sua confidenza che stavolta, a differenza di quello che avveniva per gli itemset, è già espressa in forma percentuale.

Supponendo di utilizzare come valore per la confidenza 0.5 verranno analizzati tutti gli itemset frequenti e verranno estratte solo le regole corrisponenti alle coppie (antecedenti, conseguenti) dove i conseguenti sono presenti insieme agli antecedenti in almeno metà degli itemset. Si è scelto come valore di riferimento 0.5 poiché inizialmente erano stati usati valori più bassi (per esempio 0.01 o 0.05) e sia il numero delle regole che il tempo impiegato dall'algoritmo per generarle (nonostante l'utilizzo del cluster del Politecnico) era troppo elevato.

Come per gli itemset, prima di visualizzare i dati come nell'immagine è stata attuata un'ulteriore trasformazione volta a rendere più leggibile il risultato. In questo caso la trasformazione è consistita nel separare in modo visivo antecedenti e conseguenti delle regole, lasciando solo gli antecedenti all'interno delle parentesi tonde e mettendo in evidenza il valore relativo alla confidenza della regola.

Le regole non sono ordinate ancora in nessun modo e non vi è nessun controllo o vincolo in questa fase sul numero di antecedenti o conseguenti che una regola può o deve avere, cosa che invece avverrà successivamente.

Dopo che gli itemset e le regole sono estratte è possibile accedere al loro contenuto tramite le funzioni dell'oggetto FPGrowthModel creato in precedenza freqitemset() e associationRules().

3.4.3 Filtering itemset

La maggior parte del lavoro di processamento dei dati è stato fatto dopo l'estrazione degli itemset e delle regole. I motivi sono diversi: in primo luogo il numero di risultati ottenuti è stato molto elevato e, per poter fare un'analisi corretta e capire se il programma funzionava correttamente, è stato necessario filtrare ciò che era stato ottenuto in precedenza. La ragione tuttavia più importante per questa scelta è

che alcune delle regole e degli itemset estratti non avevano senso dal punto di vista temporale e logico. Supponendo di avere per esempio una regola:

1_1_full, 2_3_full -> 0_5_empty

è possibile notare come avere nel conseguente un istante temporale (0 in questo caso) più piccolo rispetto agli istanti temporali contenuti negli antecedenti sia inutile al fine di effettuare una predizione sul comportamento e gli stati futuri delle stazioni.

Allo stesso modo si possono fare delle considerazioni sui risultati ottenuti dopo l'estrazione degli itemset frequenti, in particolare è importante considerare il fatto che se si vuole effettuare un'analisi su una finestra temporale rappresentata da degli indici numerici è necessario avere nell'elenco degli stati dell'itemset almeno uno stato dove l'istante temporale associato alla misurazione sia 0, in modo da poter fare delle predizioni sensate e avere un punto di riferimento. Per questo motivo si è deciso di applicare un filtro agli itemset ottenuti dall'estrazione volto proprio a eliminare quelli che non contenevano al loro interno nessuno stato con istante temporale pari a 0. Spark permette di implementare un filtro di questo tipo in maniera molto semplice, tramite una funzione che analizza ogni record dei risultati e restituisce un valore booleano (true o false) alla funzione chiamante a seconda che la condizione specificata all'interno del filtro sia verificata o no: se il risultato è true allora il record viene tenuto all'interno dei dati, altrimenti viene scartato. Prendendo per esempio i seguenti itemset di esempio:

1_5_full, 2_7_empty, 1_6_normal
2_4_full, 0_6_normal, 1_1_normal
1_8_normal, 1_2_empty, 2_3_normal

il risultato, dopo l'applicazione del filtro, sarà:

2_4_full, 0_6_normal, 1_1_normal

Successivamente si è scelto, per una questione di semplicità e leggibilità, di inserire un'ulteriore trasformazione con l'obiettivo di ordinare gli stati delle stazioni in base all'istante temporale a cui fanno riferimento. Prendendo ad esempio il risultato precedente, dopo l'ordinamento si otterrà:

0_5_normal, 1_1_normal, 2_4_full

Questa operazione viene effettuata non mediante un filtro ma tramite una semplice trasformazione simile a quelle utilizzate nella fase di preprocessing che sfrutta un metodo di sorting sui dati.

3.4.4 Filtering regole

Se sugli itemset il lavoro di filtering si compone di una sola e semplice trasformazione, la stessa cosa non si può dire per le regole, dove invece la fase di filtering è ben più

complessa e elaborata.

Il primo filtro che viene applicato alle regole è stato introdotto nell'ottica di poter poi utilizzare le regole per effettuare delle predizioni e ha l'obiettivo di selezionare quelle che hanno un conseguente solo.

Come visto nella sottosezione precedente le regole considerate, perchè le analisi possano avere senso e si possano fare delle predizioni, devono essere strutturate in modo che l'istante temporale del conseguente sia maggiore di tutti gli istanti temporali degli antecedenti. In più, proprio come per gli itemset, anche negli antecedenti delle regole è necessario che vi sia almeno uno stato relativo a una stazione con istante temporale pari a 0. Supponendo di avere le regole:

1_5_full, 2_7_empty -> 1_6_normal, 2_3_full
2_4_full, 0_6_normal -> 1_1_normal
0_8_normal, 1_2_empty -> 2_3_normal

il risultato, dopo l'applicazione dei filtri descritti sopra, sarà:

0_8_normal, 1_2_empty -> 2_3_normal

Nel filtering delle regole è stata inserita la possibilità di aggiungere due filtri ulteriori: il primo con lo scopo di ordinare gli istanti temporali di antecedenti e conseguente in ordine crescente, in modo da avere delle regole nella forma:

0_8_normal, 1_2_empty -> 2_3_normal
0_1_empty -> 2_5_normal

dove cioè anche gli antecedenti sono ordinati in base all'istante temporale; il secondo, invece, permette di scartare le regole in cui l'identificativo associato alla stazione nel conseguente è uguale ad almeno uno degli identificativi degli stati presenti negli antecedenti della regola. Nel caso delle seguenti regole dunque:

0_8_normal, 1_2_empty -> 2_2_normal
0_1_empty -> 2_5_normal
0_6_full, 1_3_empty -> 2_6_normal

il programma selezionerebbe solo:

0_1_empty -> 2_5_normal

Gli ultimi due filtri sono applicabili tramite un flag che può valere 0 o 1 e che viene specificato dall'utente come parametro dell'applicazione.

3.4.5 Join itemset e regole

Nei risultati visti fino a questo punto le informazioni relative agli itemset e alle regole sono separate, questo determina che anche il supporto dei primi e la confidenza

delle seconde appaiono in due output diversi. Per ottenere le informazioni congiunte il programma svolge un'operazione di Join tra i due risultati ottenuti precedentemente, andando a verificare quali sono i match tra gli itemset e le regole considerate come insieme di antecedenti e conseguente. Vengono considerati tutti gli itemset, tutte le regole e creati dei record solo dove gli antecedenti e il conseguente di una regola sono interamente contenuti in uno degli itemset. Supponendo di avere per esempio una regola del tipo:

0_1_full, 1_3_empty -> 2_5_normal 0.7

essa verrebbe unita tramite Join a un itemset:

0_1_full, 1_3_empty, 2_5_normal 2389

per generare un output:

0_1_full, 1_3_empty -> 2_5_normal 2389 0.7

Per svolgere l'operazione di Join in Spark è sufficiente utilizzare la funzione join() su uno dei due dataset passando come parametro l'altro con cui si vuole effettuare l'operazione. Il risultato finale sarà simile a quello illustrato in figura 3.13

```
Itemset: (0_10_full,1_11_full,2_9_full) Rule: 0_10_full, 1_11_full  2_9_full  0.8853005307113178 Supp
orto: 21519
Itemset: (0_7_full,1_8_full) Rule: 0_7_full  1_8_full  0.7220141489804411 Supporto: 19085
Itemset: (0_11_full,1_9_full) Rule: 0_11_full  1_9_full  0.5895209147488782 Supporto: 24438
Itemset: (0_3_empty,1_11_full) Rule: 0_3_empty  1_11_full  0.5930607676608072 Supporto: 18016
Itemset: (0_8_full,1_7_full) Rule: 0_8_full  1_7_full  0.7462575144395113 Supporto: 18993
Itemset: (0_1_empty,1_4_empty) Rule: 0_1_empty  1_4_empty  0.5957699933906146 Supporto: 18028
Itemset: (0_10_full,1_11_full) Rule: 0_10_full  1_11_full  0.6999049785481873 Supporto: 24307
Itemset: (0_9_full,0_11_full,1_10_full) Rule: 0_9_full, 0_11_full  1_10_full  0.8762628303564213 Supp
orto: 21684
Itemset: (0_10_full,1_9_full,2_11_full) Rule: 0_10_full, 1_9_full  2_11_full  0.7508613189490169 Supp
orto: 21576
Itemset: (0_9_full,1_11_full,2_10_full) Rule: 0_9_full, 1_11_full  2_10_full  0.8832406347129069 Supp
orto: 21597
Itemset: (0_3_empty,1_4_empty) Rule: 0_3_empty  1_4_empty  0.7292448482454408 Supporto: 22153
Itemset: (0_10_full,0_11_full,1_9_full) Rule: 0_10_full, 0_11_full  1_9_full  0.8798180415092807 Supp
orto: 21662
Itemset: (0_4_empty,1_3_empty) Rule: 0_4_empty  1_3_empty  0.6819118506443577 Supporto: 22171
```

Figura 3.13. Esempio di risultato ottenuto dopo l'operazione di Join su itemset e regole.

Anche in questo caso è stata introdotta un'ulteriore trasformazione per rendere più leggibile il risultato, nella versione finale dei dati scompaiono le scritte "itemset", "Rule", "Supporto" e le parentesi tonde, lasciando i dati nel formato essenziale descritto precedentemente.

3.4.6 Sorting del risultato

A questo punto dell'algoritmo i dati sono quasi pronti per essere visualizzati e nel formato corretto. Prima però di utilizzarli per l'applicazione che si occuperà di mostrare il risultato, viene effettuata un'ultima trasformazione su di essi, con l'obiettivo di ordinarli in modo da poter, successivamente, scegliere le regole più adatte per le predizioni o gli esperimenti. Il sorting realizzato ordina le regole in base alla loro confidenza, in ordine decrescente e, nel caso due regole abbiano la stessa confidenza, le ordina in base al loro supporto. L'operazione di sorting viene effettuata nello stesso modo in cui era stata effettuata precedentemente per ordinare gli istanti temporali all'interno degli itemset, utilizzando una funzione che sfrutta un metodo di sorting sui dati specificato come implementazione della funzione stessa. Il risultato finale sarà simile a quello illustrato in figura 3.14:

```
0_10_full, 0_11_full  1_9_full  0.8601342845153168  0.2248858948477135
0_9_full, 0_11_full  1_10_full  0.8577002139530981  0.22431536908628105
0_10_full, 0_11_full, 1_11_full  2_9_full  0.8539785150679723  0.19711665057491443
0_9_full, 0_11_full, 1_11_full  2_10_full  0.8522390909958942  0.195854910910208
0_10_full, 1_11_full  2_9_full  0.8433492904770046  0.21648161151584305
0_4_empty, 1_7_normal, 0_7_normal  2_8_normal  0.8395962544083668  0.15149653295883436
0_9_full, 1_11_full  2_10_full  0.835914626384351  0.21614149038883526
0_1_empty, 1_9_full  2_10_full  0.8290071770334928  0.15207803036952514
0_10_full, 0_7_normal  1_9_full  0.8278193807980329  0.16252304046344246
0_1_empty, 1_7_normal, 0_7_normal  2_8_normal  0.8265199161425576  0.15572061792328623
0_10_full, 0_11_full  2_9_full  0.8258078052874528  0.2159110857544106
0_12_normal, 0_5_normal  1_6_normal  0.8231757632912885  0.15323005354164837
0_1_empty, 0_9_full  1_10_full  0.8218370264474625  0.1513758448169929
0_4_empty, 1_4_empty, 1_7_normal  2_8_normal  0.8181333017077799  0.1513758448169929
0_9_full, 0_11_full  2_10_full  0.8175944959516718  0.21382647239533045
0_1_empty, 1_8_normal, 0_8_normal  2_7_normal  0.8162596783799881  0.15036645308522778
0_5_empty, 0_7_normal  1_8_normal  0.8150696061968474  0.16509040638988853
0_5_empty, 1_7_normal  2_8_normal  0.8146269633507853  0.1638835249714737
0_10_full, 0_8_normal  1_9_full  0.8115998879237881  0.15890239620819802
0_10_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8107971745711403  0.1586829632230317
0_4_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8094640323108883  0.16272053015009216
0_12_empty, 1_7_normal  2_8_normal  0.8074004353197247  0.1505858860703941
0_1_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8073453264292959  0.1659352233827789
0_10_full, 1_7_normal  2_9_full  0.8072481083233771  0.15567673132625295
0_11_full, 1_8_normal, 0_8_normal  2_7_normal  0.8068856351404828  0.15659834986395155
0_10_full, 0_7_normal  2_9_full  0.80675086621213811  0.15838672869305714
```

Figura 3.14. Esempio di risultato ottenuto dopo l'operazione di sorting.

Nel risultato mostrato gli antecedenti sono separati mediante virgole e il conseguente è separato da essi tramite una coppia di spazi. I due valori alla destra di ogni regola rappresentano rispettivamente la confidenza della regola e il supporto dell'itemset associato (trasformato in supporto percentuale dividendo il suo valore per il numero di record totali).
Dopo quest'ultima sottofase la fase di processing termina e i dati sono pronti per essere visualizzati dall'utente.

3.5 Visualizzazione delle regole

I dati e i risultati descritti finora sono dati in formato testuale, utili all'utente o allo sviluppatore per capire che il programma funzioni, ma non sufficientemente intuitivi da utilizzare se si vuole avere un'idea della posizione geografica delle stazioni coinvolte e della distribuzione delle regole. Per questo motivo, durante il periodo di tesi, si sono cercate diverse soluzioni per fornire una visualizzazione grafica dei risultati ottenuti dopo la fase di processing dei dati. In particolare sono state pensate e sviluppate due modalità di fruizione grafica: la prima sfrutta il linguaggio KML e la possibilità che alcuni tool online offrono di caricare dei file in formato .kml per la visualizzazione di informazioni spaziali, la seconda invece consiste in un'applicazione grafica sviluppata in JavaFX appositamente per l'argomento di tesi. L'idea alla base di entrambe è quella di visualizzare su una mappa 2D i punti dove sono collocate le stazioni, dando un'indicazione sulla regola di cui fanno parte, la sua confidenza e il supporto dell'itemset associato.

3.5.1 Creazione file KML

La prima delle due soluzioni proposte sfrutta il linguaggio KML. L'applicazione realizzata in Spark, dopo aver processato i dati, effettua un matching tra le regole ottenute e il file di input contenente l'associazione tra l'identificativo di una stazione e le sue coordinate, andando a sostituire in ogni regola l'id corrispondente alla stazione con la sua latitudine e longitudine. Una volta fatto ciò viene creato per ogni regola un file .kml contentente un placemarker per ogni coppia di coordinare latitudine-longitudine presente nella regola. Un placermarker è un identificativo utilizzato da Google Maps e Google Earth per segnare graficamente la posizione di un punto su una mappa. Se una regola contiene tre stazioni si avranno tre coppie latitudine-longitudine e tre placemarker che corrisponderanno ad altrettanti punti nello spazio. All'interno di ogni file si è scelto di identificare, in fase di visualizzazione, con il colore verde gli antecedenti della regola e con il colore rosso il conseguente, che rappresenta una stazione in uno stato critico. Vicino a ogni placemarker è inoltre posta una label contenente lo stato della stazione e il suo identificativo. Per poter visualizzare i file è sufficiente accedere al sito di Google Earth e caricarlo tramite l'apposita sezione denominata "I miei luoghi" e cliccando su "Importa file KML":

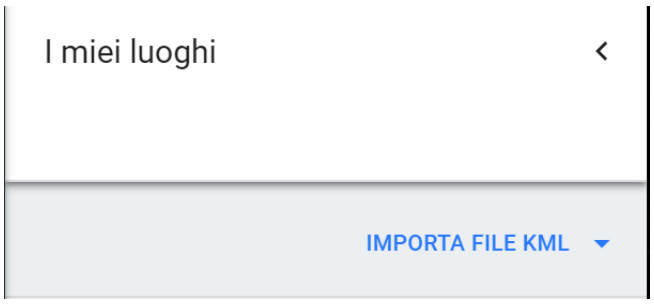

Figura 3.15. Sezione di Google Earth dove caricare il file .kml

Figura 3.16. Visualizzazione di uno dei file .kml su Google Earth

Google Earth mostra una mappa ottenuta tramite satellite della zona, andando a posizionare un placemarker per ogni stazione contenuta nella regola. Nel caso ci siano due stati relativi alla stessa stazione il placemarker sarà comunque uno solo. Vicino a ogni stazione viene inoltre inserita una label che indica lo stato di quello stazione, nel caso in cui ci siano più stati relativi alla stessa stazione si avrà una label per ogni stato, anche se il placemarker è uno solo.

Come detto precedentemente è necessario visualizzare anche le informazioni relative alla confidenza della regola e dell'itemset ad esso associato. Questo è possibile cliccando su uno qualsiasi dei placemarker che Google Earth mostra. Una volta fatto questo viene mostrato all'utente un pannello contenente due righe: sulla prima vi è l'informazione sul supporto e sulla seconda quella sulla confidenza. Queste informazioni sono state inserite tramite codice nel file kml, in un modo simile a quello utilizzato per specificate il colore dei placemarker, andando cioè a inserire, siccome il KML è un linguaggio di markup, dei tag appositi (il tag <color> nel primo caso e il tag <options> nel secondo).

Figura 3.17. Pannello con informazioni relative a supporto e confidenza della regola visualizzate

I file kml creati dal programma vengono salvati in una cartella il cui nome è specificato come parametro del programma, mentre il nome di ogni file è generato automaticamente in base a una numerazione progressiva: il primo file avrà nome "file1", il secondo "file2" etc.. L'utilizzo di questi file, tuttavia, non è molto user-friendly, in particolare con questa modalità non è possibile avere una visione complessiva delle regole e delle stazioni critiche. Inoltre, per passare da una regola a un'altra è necessario uscire da Google Earth, selezionare un altro file e ricaricarlo, il che rende la visualizzazione delle regole lunga, pesante e complessa. Questi sono i motivi per cui si è deciso di realizzare un'applicazione stand alone in JavaFX che permettesse di dare all'utente una esperienza più semplice e intuitiva.

3.5.2 Visualizer

L'applicazione grafica che tratteremo in questa sottosezione e che d'ora in poi chiameremo per semplicità Visualizer è stata sviluppata con il linguaggio JavaFX e la libreria open source GMapsFX. Il Visualizer si compone di una finestra al cui interno viene caricata una mappa di Google Maps e di alcuni bottoni nella sua parte inferiore che servono a controllare le azioni dell'utente. La mappa di Google Maps viene caricata all'avvio dell'applicazione grazie alla libreria GMapsFX e presenta la possibilità di zoomare e spostarsi lungo la mappa tramite mouse. Un esempio dell'interfaccia è il seguente:

Figura 3.18. Esempio interfaccia Visualizer

Il Visualizer, per funzionare correttamente, ha bisogno di tre file il cui percorso viene specificato dall'utente nelle caselle di testo in basso a sinistra:

1. File contentente i record delle misurazioni delle stazioni
2. File contente le regole con relativa confidenza e supporto dopo la fase di processing dei dati
3. File contente l'associazione tra id delle stazioni e loro coordinate

Una volta inseriti i percorsi di questi file è possibile iniziare a usare l'applicazione. Il Visualizer può essere utilizzato in due modalità principali: la prima permette di visualizzare le regole una per volta, mostrando anche la loro confidenza e il loro supporto, mentre la seconda ha l'obiettivo di dare una visione più generale all'utente di quello che succede mostrando tutti i conseguenti delle regole.

Si accede alla modalità di visualizzazione singola cliccando sul bottone S vicino alle caselle di testo. Una volta fatto ciò l'interfaccia cambia leggermente mostrando, in alto, una serie di label e, sotto di esse, due bottoni intervallati dalla regola considerata con il relativo supporto e confidenza. Cliccando sui bottoni è possibile cambiare regola da visualizzare. Per ogni regola l'applicazione mostra un placemarker per ogni stazione coinvolta al cui centro è indicato l'identificativo della stazione. Gli stati delle stazioni sono indicati nella regola posta nella label tra i due bottoni mentre sopra di essa si hanno le informazioni relative al supporto e la confidenza. Di seguito un esempio della schermata:

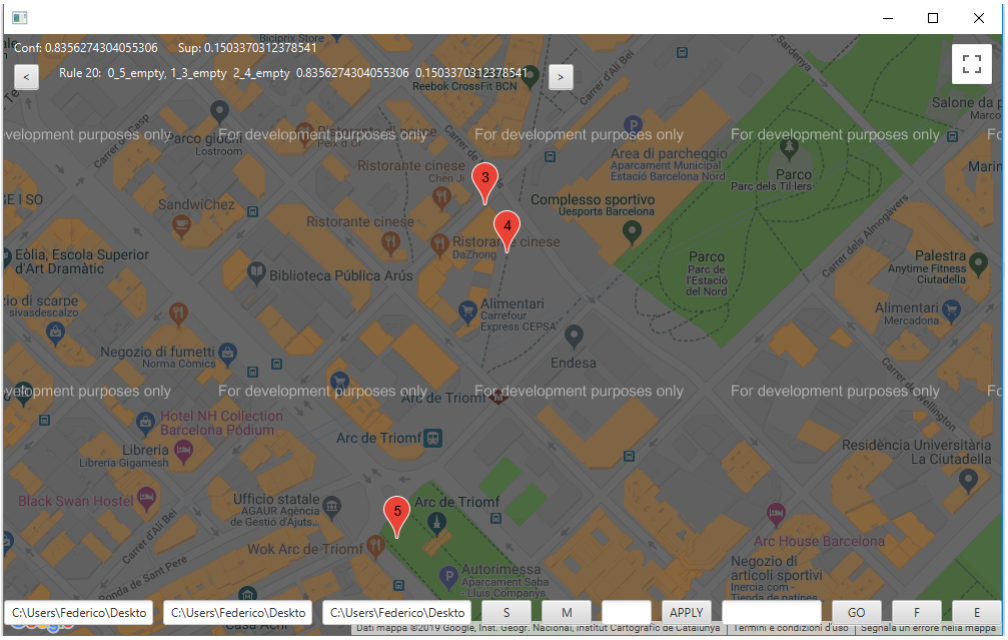

Figura 3.19. Esempio interfaccia modalità di visualizzazione singola delle regole

È possibile inoltre specificare, grazie alla casella di testo vicino al tasto APPLY, un valore di confidenza minimo in modo da visualizzare singolarmente solo le regole con confidenza maggiore o uguale a quella specificata. Questo grazie al fatto che le regole sono inserite, all'avvio dell'applicazione, in una struttura dati sulla quale è possibile operare facilmente delle operazioni di sorting e filtering.

Premendo sul bottone M è invece possibile accedere alla modalità di visualizzazione multipla delle regole. Anche in questo caso l'interfaccia cambia leggermente: spariscono le label e i bottoni precedenti (se si era in modalità singola) e al loro posto appare uno slider che permette all'utente di selezionare un valore di confidenza minima da usare come soglia. Il valore scelto viene usato dal programma per filtrare le regole e scegliere di visualizzare solo i conseguenti delle regole la cui confidenza è maggiore o uguale a quella specificata dall'utente. Ogni volta che il valore dello slider cambia l'applicazione ridisegna tutti i conseguenti, ricalcolando quali regole superano la soglia di confidenza. L'utente può scegliere di passare in qualsiasi momento dalla modalità di visualizzazione singola a quella di visualizzazione multipla, il programma annulla le informazioni precedenti e calcola quelle richieste ogni volta che viene premuto un bottone. Il passaggio da una modalità all'altra è gestito tramite flag, uno per ogni modalità, che permettono di capire all'applicazione in quale stato si trova l'utente. Inizialmente si era pensato di disegnare, nella modalità multipla anche gli antecedenti di tutte le regole, ma il risultato risultava confusionario e poco leggibile, per questo motivo si è optato successivamente per la soluzione descritta sopra. Un esempio di interfaccia di visualizzazione multipla delle regole è il seguente:

Figura 3.20. Esempio interfaccia modalità di visualizzazione multipla delle regole

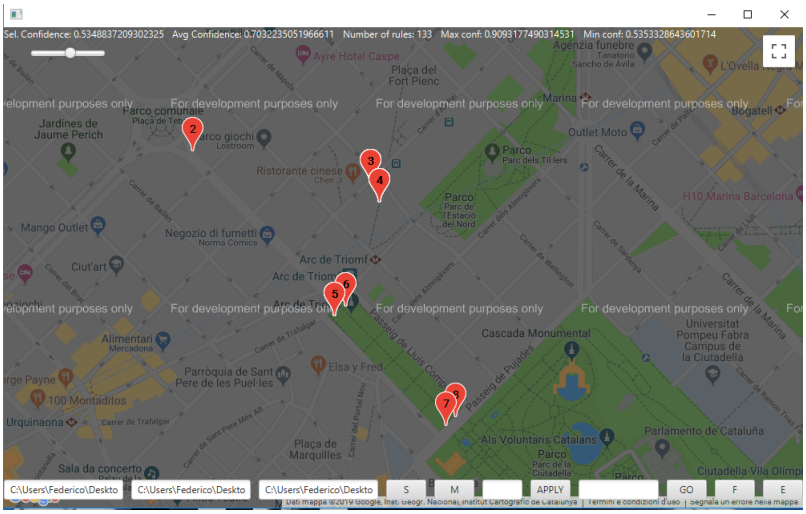

Figura 3.21. Esempio interfaccia modalità di visualizzazione multipla delle regole

Le label posizionate sopra lo slider mostrano all'utente alcune informazioni utili all'analisi complessiva delle regole. Oltre alla soglia minima di confidenza indicata dall'utente vi sono anche la confidenza media di tutte le regole che superano la soglia, il numero di regole con confidenza maggiore o uguale alla soglia posta dall'utente, la confidenza minima e massima delle regole visualizzate. Cambiando il valore dello slider anche queste label si modificano, mostrando i valori aggiornati:

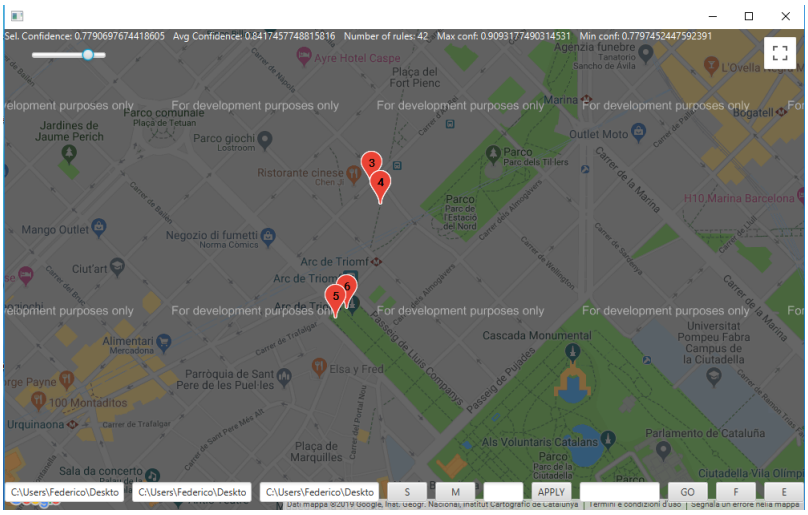

Figura 3.22. Esempio interfaccia modalità di visualizzazione multipla delle regole

L'ultima feature che il Visualizer implementa è quella di poter prevedere lo stato delle stazioni dopo un istante temporale. Specificando nella casella di testo vicino al pulsante GO l'orario concatenato alla data che si vuole analizzare e la durata in minuti di un istante temporale (separati da una virgola), il programma verifica, per tutte le regole, che i suoi antecedenti siano contenuti nel file dei record e, in caso affermativo, mostra il conseguente all'utente segnandone lo stato. I due pulsanti F ed E, inoltre, funzionano come filtro e permettono all'utente di visualizzare solo le stazioni che si trovano in un determinato stato critico. Premendo infatti il tasto F l'applicazione mostrerà all'utente solo le stazioni che si trovano in uno stato full, mentre, nel caso fosse premuto il tasto E, verranno visualizzate solo le stazioni in uno stato empty. Supponendo quindi di avere una regola:

0_1_full, 1_3_empty -> 2_5_empty 2389 0.7

e di inserire all'interno della casella di testo: "20:30_05/06/2008,2", il programma considererà l'istante temporale 1 come quello presente (si sceglie l'istante temporale più alto tra quelli degli antecedenti) e se nel file dei record saranno presenti le seguenti misurazioni:

20:28_05/06/2008 1 full

la stazione 5 verrà considerata come empty e il suo placermarker visualizzato sulla mappa. Di seguito alcune immagini del funzionamento di questa feature:

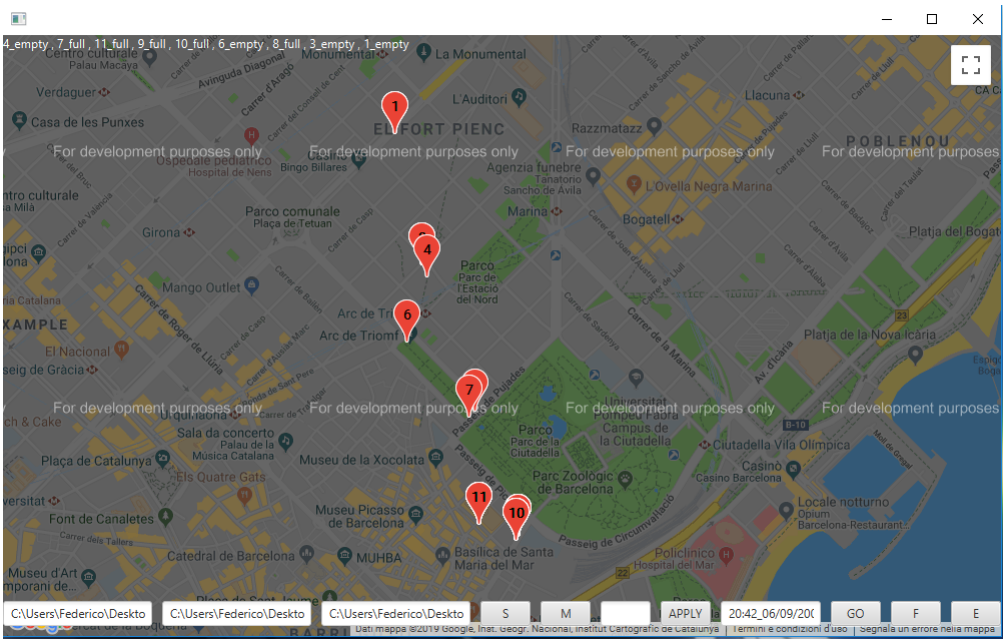

Figura 3.23. Esempio predizione stazioni critiche

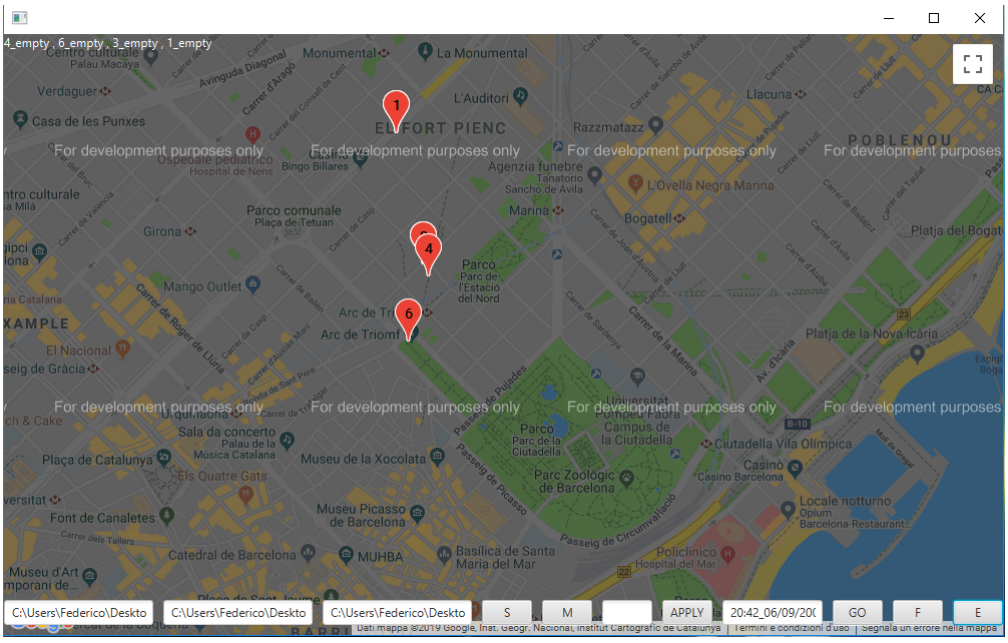

Figura 3.24. Esempio predizione stazioni critiche full

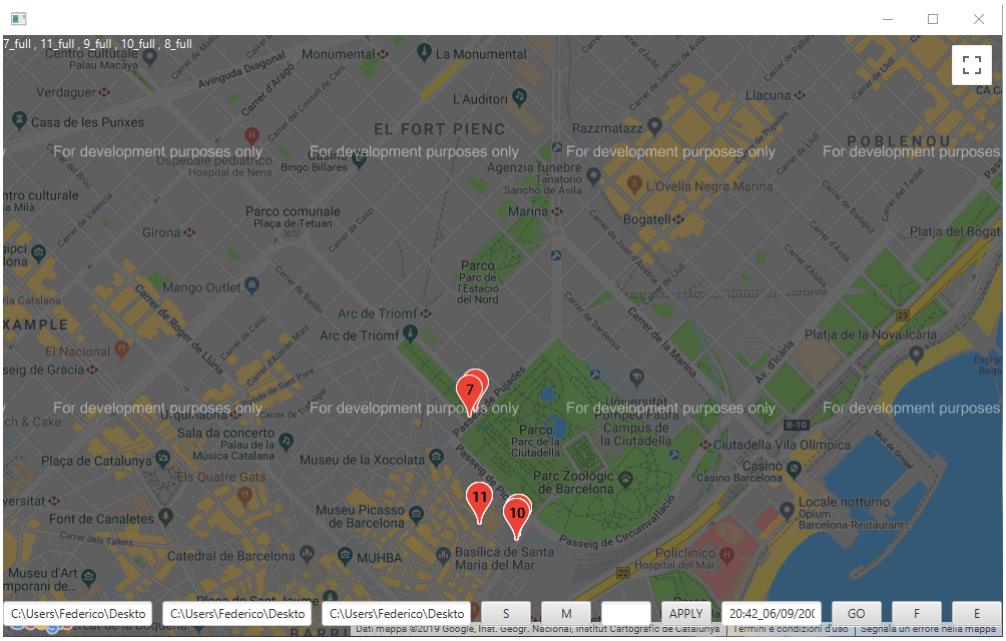

Figura 3.25. Esempio predizione stazioni critiche empty

3.6 Algoritmo di predizione

Per calcolare i valori delle predizioni, l'applicazione sfrutta due file di input del Visualizer, il primo contiene l'elenco delle regole, mentre il secondo registra, per ogni riga, l'elenco degli stati di tutte le stazioni in una determinata finestra temporale. Come descritto nel capitolo precedente, infatti, a un certo punto dell'elaborazione e del preprocessing dei dati, le informazioni relative agli stati delle stazioni vengono raggruppate utilizzando l'istante temporale come chiave. Il risultato di questa elaborazione viene salvato su un file e utilizzato dal Visualizer per fare le previsioni. Si è scelto di utilizzare questo metodo poiché l'alternativa (che è stata anche testata inizialmente) a questo procedimento è utilizzare il file di partenza dove sono contenuti, per ogni riga, i record misurati dai sensori, ma questo determinava un tempo di elaborazione delle regole estremamente più lungo andando a scorrere il file molteplici volte (si ricorda che l'applicazione del Visualizer non fa uso di Apache Spark, ma è un'applicazione stand alone sviluppata appositamente per la visualizzazione delle regole).

```
0_7_full 1_7_full 2_7_full 0_8_full 1_8_full 2_8_full 0_9_normal 1_9_normal 2_9_normal 0_10_full 1_10
_full 2_10_full 0_11_full 1_11_full 2_11_full 0_12_normal 1_12_normal 2_12_normal 0_1_normal 1_1_norm
al 2_1_normal 0_2_full 1_2_full 2_2_full 0_3_normal 1_3_normal 2_3_normal 0_4_normal 1_4_normal 2_4_n
ormal 0_5_full 1_5_full 2_5_full 0_6_normal 1_6_normal 2_6_normal
0_7_empty 1_7_empty 2_7_empty 0_8_empty 1_8_empty 2_8_empty 0_9_empty 1_9_empty 2_9_empty 0_10_empty
 1_10_empty 2_10_empty 0_11_empty 1_11_empty 2_11_empty 0_12_empty 1_12_empty 2_12_empty 0_1_full 1_1
_full 2_1_full 0_2_normal 1_2_normal 2_2_normal 0_3_empty 1_3_empty 2_3_empty 0_4_empty 1_4_empty 2_4
_empty 0_5_normal 1_5_normal 2_5_normal 0_6_empty 1_6_empty 2_6_empty
0_7_empty 1_7_normal 2_7_normal 0_8_normal 1_8_normal 2_8_normal 0_9_empty 1_9_empty 2_9_empty 0_10_e
mpty 1_10_empty 2_10_empty 0_11_normal 1_11_normal 2_11_normal 0_12_normal 1_12_normal 2_12_normal 0_
1_empty 1_1_empty 2_1_empty 0_2_empty 1_2_empty 2_2_empty 0_3_empty 1_3_empty 2_3_empty 0_4_empty 1_4
_empty 2_4_empty 0_5_empty 1_5_empty 2_5_empty 0_6_empty 1_6_empty 2_6_empty
0_7_full 1_7_full 2_7_normal 0_8_normal 1_8_normal 2_8_normal 0_9_empty 1_9_empty 2_9_empty 0_10_empt
y 1_10_empty 2_10_empty 0_11_normal 1_11_normal 2_11_normal 0_12_normal 1_12_normal 2_12_normal 0_1_n
ormal 1_1_normal 2_1_normal 0_2_normal 1_2_normal 2_2_normal 0_3_empty 1_3_empty 2_3_empty 0_4_empty
```

Figura 3.26. File contenente gli stati delle stazioni raggruppati per istante temporale

```
0_10_full, 0_11_full  1_9_full  0.8601342845153168  0.2248858948477135
0_9_full, 0_11_full  1_10_full  0.8577002139530981  0.224315369086281O5
0_10_full, 0_11_full, 1_11_full  2_9_full  0.8539785150679723  0.19711665057491443
0_9_full, 0_11_full, 1_11_full  2_10_full  0.8522390909958942  0.195854910910208
0_10_full, 1_11_full  2_9_full  0.8433492904770046  0.21648161151584305
0_4_empty, 1_7_normal, 0_7_normal  2_8_normal  0.8395962544083668  0.15149653295883436
0_9_full, 1_11_full  2_10_full  0.835914626384351  0.216141490388835526
0_1_empty, 1_9_full  2_10_full  0.8290071770334928  0.15207803036952514
0_10_full, 0_7_normal  1_9_full  0.8278193807980329  0.16252304046344246
0_1_empty, 1_7_normal, 0_7_normal  2_8_normal  0.8265199161425576  0.15572061792328623
0_10_full, 0_11_full  2_9_full  0.8258078052874528  0.2159110857544106
0_12_normal, 0_5_normal  1_6_normal  0.8231757632912885  0.153230053541648337
0_1_empty, 0_9_full  1_10_full  0.8218370264474625  0.1513758448169929
0_4_empty, 1_4_empty, 1_7_normal  2_8_normal  0.8181333017077799  0.1513758448169929
0_9_full, 0_11_full  2_10_full  0.8175944959516718  0.21382647239533045
0_1_empty, 1_8_normal, 0_8_normal  2_7_normal  0.8162596783799881  0.15036645308522778
0_5_empty, 0_7_normal  1_8_normal  0.8150696061968474  0.16509040638988853
0_5_empty, 1_7_normal  2_8_normal  0.8146269633507853  0.1638835249714737
0_10_full, 0_8_normal  1_9_full  0.8115998879237881  0.15890239620819802
0_10_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8107971745711403  0.1586829632230317
0_4_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8094640323108883  0.16272053015009216
0_12_empty, 1_7_normal  2_8_normal  0.8074004353197247  0.1505858860703941
0_1_normal, 1_5_normal, 0_5_normal  2_6_normal  0.8073453264292959  0.1659352233827789
0_10_full, 1_7_normal  2_9_full  0.8072481083233771  0.15567673132625295
0_11_full, 1_8_normal, 0_8_normal  2_7_normal  0.8068856351404828  0.15659834986395155
0_10_full, 0_7_normal  2_9_full  0.8067508662121381  0.15838672869305714
```

Figura 3.27. File contenente le regole utilizzato per le previsioni

L'algoritmo di analisi delle previsioni controlla, per ogni id delle stazioni, tutte le regole e, per ciascuna di esse, sceglie l'istante temporale più grande tra i precedenti. A questo punto controlla, per ogni riga, se nel file contenente gli stati delle stazioni raggruppati sono contenuti tutti gli antecedenti della regola (eventualmente shiftati per prevedere l'istante temporale 2) e, se questo accade, allora il conseguente della regola viene scelto come previsione. Per velocizzare l'algoritmo le regole sono state ordinate in base a confidenza e supporto, in modo che alla prima regola trovata l'algoritmo possa passare a esamina l'id della stazione successivo.

Supponendo di avere la regola:

0_1_full, 1_2_empty -> 2_3_normal

l'algoritmo controlla che nel file siano contenuti, nelle righe:

1_2_empty

0_1_full

Nel caso invece di una regola del tipo:

0_1_full -> 1_3_empty

viene cercato nel file lo stato:

1_1_full

perché si vuole predire l'istante temporale 2.

Dopo aver controllato questo viene anche verificato se la previsione è corretta o no andando a controllare che sia contenuto anche il conseguente nello stesso record del file e se la criticità coincide o no con quella predetta.

```
for every record in list of records

        declare vector of string from record splitted by char " "
        declare list of values
        for every string of the vector
              add string of the vector to string of values

        for every station from 1 to 12
              declare selected rule

              for every rule in list of rules
                   declare countRule
                   if last point of the rule has the same index of the station
                       declare deltaTime ad the difference between temporalWindow value and temp index of point
                   for every point of the rule
                       check if list of values contains antecedents of the rule and increment counter

              if countRule equals to number of antecedents of the checked rule
                   selectedRule = checkedRule
```

Figura 3.28. pseudocodice dell'algoritmo di predizione

Capitolo 4

Esperimenti

4.1 Formato dei dati

Prima di procedere con la descrizione degli esperimenti che si sono realizzati dopo lo sviluppo dell'algoritmo di tesi è doveroso specificare il formato dei dati su cui si è lavorato. I dati utilizzati sono relativi a delle stazioni di biciclette di Barcellona, ciascuna delle quali può avere fino a un massimo di 18 posti occupati. Come spiegato nel capitolo precedente, all'inizio della fase di preprocessing dei dati le informazioni in più presenti nei dati in input vengono trasformate nell'evento considerato. In questo caso gli eventi sono relativi agli stati delle stazioni e, a seconda di una soglia inserita dall'utente come parametro dell'applicazione, una stazione può trovarsi in uno stato full, empty o normal a seconda che i posti liberi o occupati siano minori della soglia fissata. I dati di partenza sono stati ottenuti tramite dei sensori che hanno misurato ogni due minuti il numero di posti liberi e il numero di posti occupati per ogni stazione durante un periodo di cinque mesi, da Maggio a Settembre del 2008, con delle pause nelle misurazioni in orari notturni (più precisamente dalle 00:00 alle 05:00).

Il risultato di queste misurazioni sono stati due file testuali. Il primo contiene al suo interno, per ogni riga, un record che indica l'id della stazione, l'orario della misurazione, la data della misurazione, il numero di posti occupati e il numero di posti liberi per quella stazione. Il formato è il seguente:

1 2008-05-15 12:01:00 0 18

Il secondo file ottenuto, invece, contiene per ogni riga l'associazione tra l'id della stazione, il nome e le sue coordinate geografiche, utili successivamente per la visualizzazione delle regole. Il suo formato è simile al precedente:

1 41.397978 2.180019 Gran Via Corts Catalanes

Siccome il formato dei dati è testuale, la strategia che si è scelto di adottare per la loro analisi è stata quella di processare riga per riga i file, separando i singoli campi grazie ai caratteri speciali che li delimitavano (principalmente spazi e TAB) per poi creare delle strutture dati adatte al lavoro da svolgere.

Il formato testuale è stato adottato anche per i dati forniti in output dal programma sviluppato, dati che vengono anche utilizzati, come è stato spiegato precedentemente, per la visualizzazione grafica del risultato.

Nonostante il caso studio preso in esame in questa tesi, è bene specificare che l'algoritmo realizzato è applicabile a qualsiasi dataset che si riferisca a dati di tipo spazio-temporale il cui formato sia simile a quello dei dati descritti nel capitolo precedente.

4.2 Fase di sperimentazione

La seconda metà del periodo di tesi è stato utilizzato per la fase di sperimentazione. In particolare, il candidato ha cercato di capire come alcuni valori dell'algoritmo sviluppato influenzassero i risultati, dal numero di regole generate alla loro composizione, fino a osservare come la predizione delle stazioni critiche varia in base ad alcuni di essi. Per velocizzare l'esecuzione dell'algoritmo e aumentare il numero di esperimenti è stato utilizzato per ognuno di essi il cluster del Politecnico di Torino, a cui si è acceduto tramite un'interfaccia web chiamata Hue che permette di visualizzare i file caricati sul file system distribuito (HDFS) e l'output dell'algoritmo.

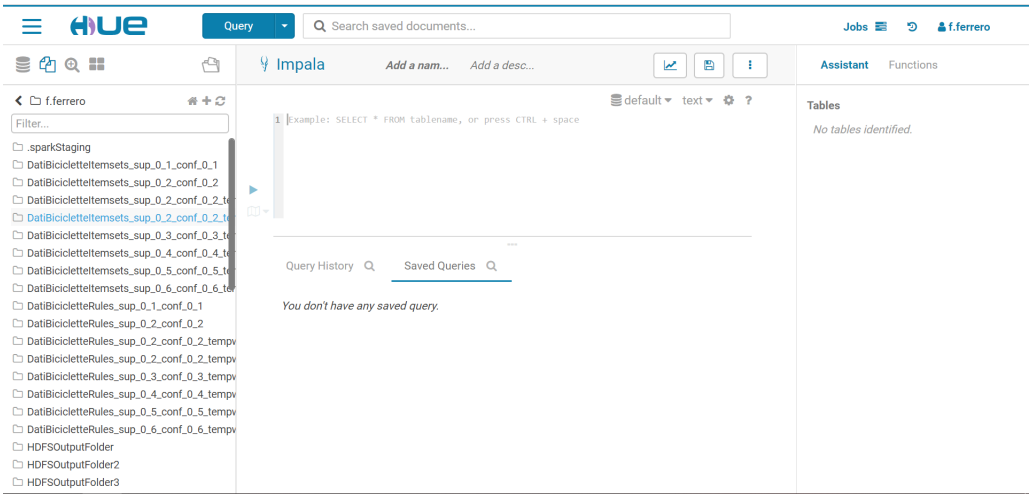

Figura 4.1. Interfaccia Hue

In questo capitolo verranno esaminati, uno per uno, tutti gli esperimenti eseguiti e i loro risultati. Verranno inoltre inseriti grafici utili a visualizzarli e a dare loro un'interpretazione.

4.3 Estrazione regole e variazione parametri

Durante la seconda parte dello svolgimento della tesi sono stati sviluppati due tipi di esperimenti: i primi volti a cambiare i parametri dell'algoritmo per osservare le caratteristiche delle regole estratte mentre i secondi con l'obiettivo di capire se le predizioni effettuate sono precise e quantificarne approssimativamente un valore. In tutti gli esperimenti della prima parte si sono utilizzati una serie di parametri ben definiti e, in ogni esperimento, due di essi vengono fatti variare mentre gli altri rimangono fissi:

1. Support: supporto minimo utilizzato per l'estrazione degli itemset frequenti
2. Confidence: confidenza minima utilizzata per l'estrazione delle regole
3. Threshold: valore utilizzato come soglia per indicare, in base al numero di posti liberi o usati, se una stazione si trova in uno stato full o empty
4. Minutes: minuti trascorsi tra un istante temporale e il seguente
5. Temporal Window: finestra temporale considerata nella generazione delle regole, indica quanti istanti temporali vengono considerati.

4.3.1 Relazione tra supporto e numero di regole

L'ultima analisi relativa alla variazione del supporto è stata fatta considerando il numero di regole in output all'applicazione. Essendo il supporto una soglia per gli itemset frequenti il risultato atteso dall'esperimento era una relazione di dipendenza nella quale all'aumentare del supporto il numero di regole diminuisse. I test hanno verificato questa ipotesi come è possibile notare dal grafico in figura 4.2.

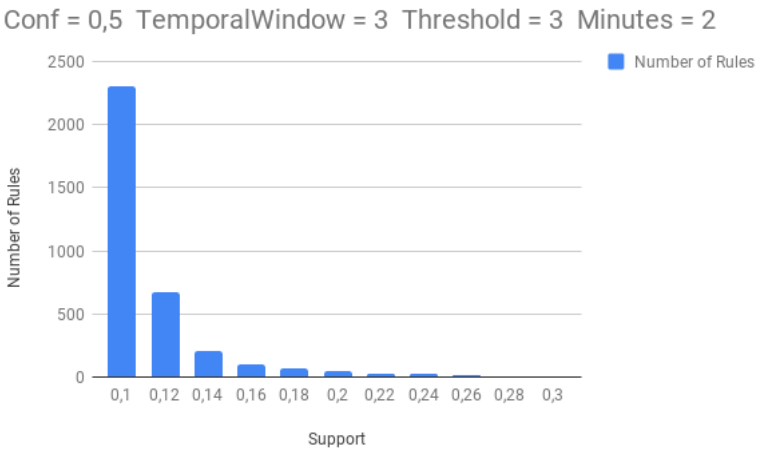

Figura 4.2. Relazione tra supporto e numero di regole 2

Anche per questo esperimento sono stati utilizzati gli stessi valori dei due precedenti. Dalle figure mostrate è possibile notare come all'aumentare del supporto il numero di regole diminuisca con un andamento esponenziale.

4.3.2 Relazione tra supporto e confidenza media delle regole

Il primo test effettuato riguarda la relazione tra il supporto e la confidenza media delle regole generate e, in particolare, come la variazione del supporto può infierire sulle caratteristiche delle regole estratte. Per questo test i valori relativi a Confidenza, Threshold, Minutes e Temporal Window sono stati mantenuti rispettivamente a 0.5, 3, 2 e 3, mentre il supporto è stato fatto variare tra un valore minimo di 0.1 e un valore massimo di 0.3, valore al di sopra del quale il numero di regole generate era troppo basso per poter analizzare i risultati. Questo per una questione di semplicità e tempo di generazione dei risultati, dato che, al variare di alcuni di questi parametri, aumenta il numero di regole generate e quindi il tempo necessario a visualizzare i risultati. La confidenza media è stata calcolata tramite una funzione che raccoglie tutti i valori di confidenza delle regole estratte e ne genera in output al programma la media.

Essendo il supporto un valore non legato al valore della confidenza e indicante semplicemente il valore minimo per cui un itemset deve essere selezionato se presente un numero di volte superiore a quel valore nei record totali, esso è indipendente dalla confidenza delle regole che vengono estratte successivamente durante la fase di rule mining. Per questo motivo il risultato aspettato prima di questo esperimento era un'indipendenza dei due parametri, quindi nessuna relazione o legame particolare.

Gli esperimenti hanno confermato questa teoria, come è possibile notare dal grafico in figura 4.3.

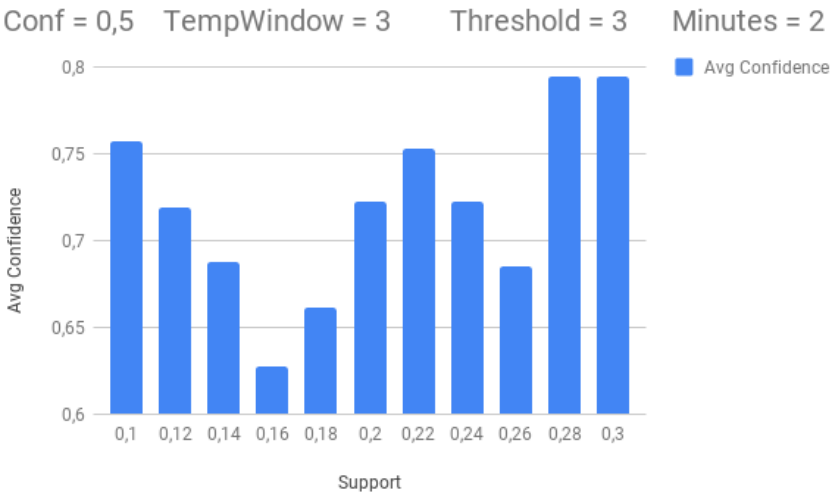

Figura 4.3. Relazione tra supporto e confidenza media delle regole 2

Come è possibile notare dalla figura mostrata, il valore della confidenza media delle regole non aumenta e non diminuisce al variare del supporto in maniera dipendente da esso. I due picchi relativi ai valori 0.28 e 0.3 di Supporto sono dovuti al ridotto numero di regole estratte, le quali avevano tutte una confidenza molto alta (pari, come mostrato, a 0.8).

4.3.3 Relazione tra supporto e lunghezza media delle regole

Un altro valore generato in output dall'applicazione, oltre alla confidenza media delle regole, è la loro lunghezza media. Con essa si intende il numero di antecedenti e conseguente che si trovano in una regole estratta. Per esempio, la regola:

0_1_full, 1_2_empty -> 2_4_full

ha lunghezza 3, perchè contiene al suo interno due antecedenti e un conseguente.

Anche per questo esperimento i valori relativi a Confidenza, Threshold, Minutes e Temporal Window sono stati mantenuti rispettivamente a 0.5, 3, 2, 3, mentre il supporto è stato fatto variare ancora una volta tra 0.1 e 0.3 a intervalli di 0.02 in ogni esperimento. Se per la confidenza media delle regole non vi era alcun legame con il supporto, per la lunghezza media sembra invece esserci una dipendenza.

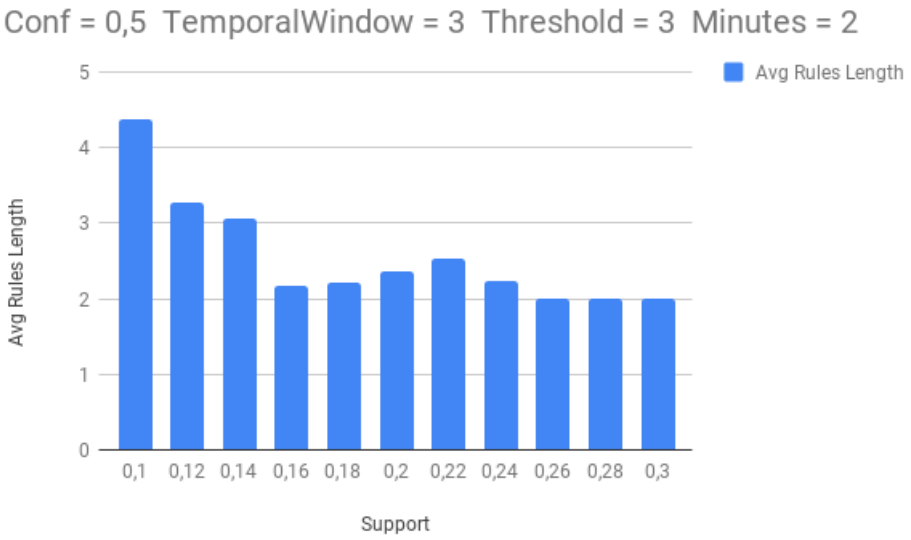

Figura 4.4. Relazione tra supporto e lunghezza media delle regole

In particolare, all'aumentare del supporto la lunghezza media delle regole tende a diminuire, generando, per la maggior parte e con questi parametri, regole composte da un solo antecedente e un solo conseguente.

4.3.4 Relazione tra confidenza e numero di regole

Come per il supporto, anche il valore della confidenza minima rappresenta una soglia per il numero di regole estratte. Per questo motivo è stato condotto un test relativo alla variazione di quest'ultima, per verificare se, come accade per il supporto, il numero di regole diminuisce all'aumentare del valore di confidenza inserito come parametro. Per questo esperimento è stato utilizzato un valore di supporto fisso pari a 0.15, mentre tutti gli altri parametri sono stati mantenuti ai valori dei test precedenti. La confidenza minima, invece, è stata fatta variare tra un minimo di 0.1 e un massimo di 1, andando a verificare dunque quante regole vengono estratte oltre una certa soglia di confidenza percentuale.

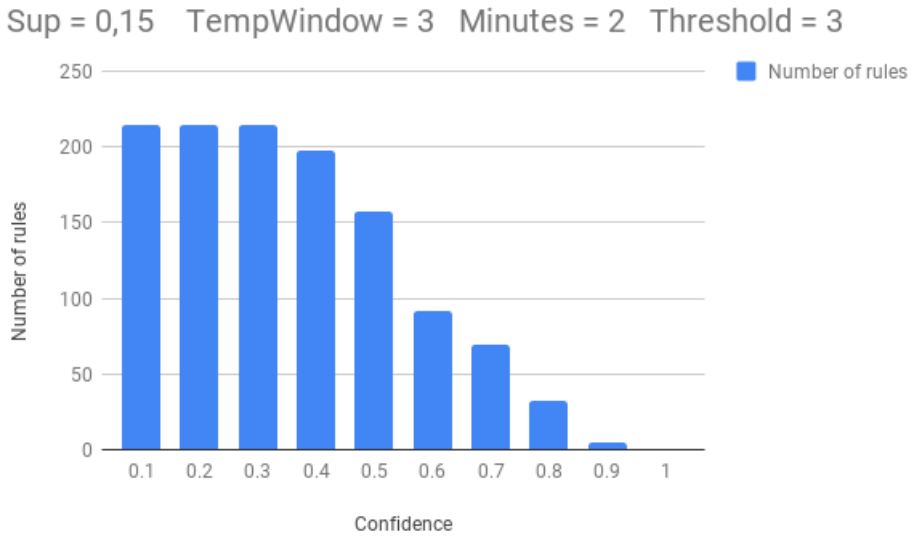

Figura 4.5. Relazione tra confidenza e numero di regole estratte

Come è possibile notare dalla figura, anche aumentando la confidenza, così come avviene per il supporto, il numero di regole estratte diminuisce, confermando le previsioni. In particolare, è interessante notare due cose: la prima è che per valori di confidenza minima relativamente bassi il numero di regole diminuisce in maniera minore rispetto a come diminuisce per valori di confidenza più alti, confermando anche in questo caso un andamento di diminuzione esponenziale; la seconda è come per un valore di confidenza minima pari a 1 non venga estratta alcuna regola. Questo significa che non vi è nessuna regola che si verifica nel 100% dei casi.

Dopo aver analizzato il supporto e la confidenza minima si è passati ai test relativi agli altri parametri dell'applicazione, facendo variare dunque a turno i minuti di un istante temporale, la soglia che definisce quando una stazione è piena o full e la finestra temporale considerata nell'estrazione delle regole, mantenendo invece costanti i valori relativi a supporto e confidenza minima.

4.3.5 Relazione tra durata dell'istante temporale e lunghezza media delle regole

Il primo degli altri parametri considerati è stato quello relativo alla durata di un istante temporale. Precedentemente i test sono stati fatti considerando la durata dell'istante temporale pari all'intervallo tra le misurazioni effettuate dai sensori che hanno fornito i dati in input (2 minuti), ma l'applicazione è pensata perché l'utente possa inserire una durata a piacimento e, per questo motivo, si è deciso di testare il comportamento dei risultati al variare di questo parametro.

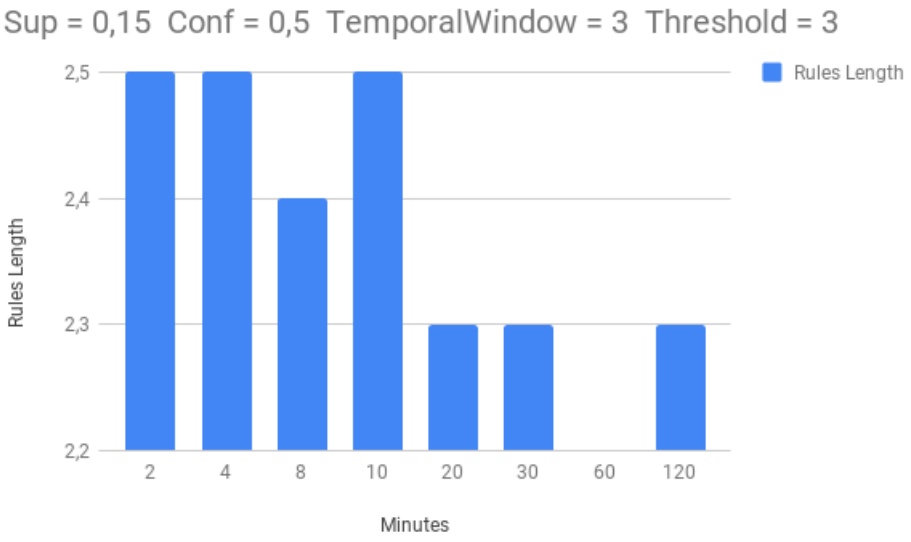

Figura 4.6. Relazione tra durata istante temporale e lunghezza media delle regole

Come è possibile notare dal grafico mostrato, non sembra esserci alcuna relazione tra i minuti di un istante temporale e la lunghezza delle regole estratte. È possibile però notare come, per valori bassi di durata, la lunghezza delle regole estratte sia maggiore rispetto a quella delle regole estratte con un valore di durata più alto. Questo probabilmente perchè la maggior parte delle regole con lunghezza maggiore a 2.5 sono relative a stazioni considerate in degli istanti temporali molto vicini tra loro e in alcuni momenti precisi del giorno (per esempio al mattino presto o la sera tardi). Supponendo per esempio di avere tre stazioni vicine identificate dai numeri 9, 10 e 11 è molto probabile che esse siano collegate in degli istanti temporali vicini rispetto a intervalli di tempo più grandi, con delle regole simili alla seguente:

0_9_full, 1_10_full -> 2_11_full

dove cioè tutti gli stati delle stazioni considerate hanno lo stesso valore.

4.3.6 Relazione tra durata dell'istante temporale e numero di regole

Come per il supporto e la confidenza, anche per la durata dell'istante temporale si sono fatti dei test per cercare di capire come al variare di quest'ultima cambiasse il numero di regole estratte. I valori utilizzati come durata sono gli stessi dell'esperimento precedente, partendo da durate temporali piccole per arrivare fino a dei valori corrispondenti a delle ore, mentre tutti gli altri parametri hanno gli stessi valori dei test precedenti. Al supporto e alla confidenza, in particolare, sono stati assegnati rispettivamente i valori 0.15 e 0.5

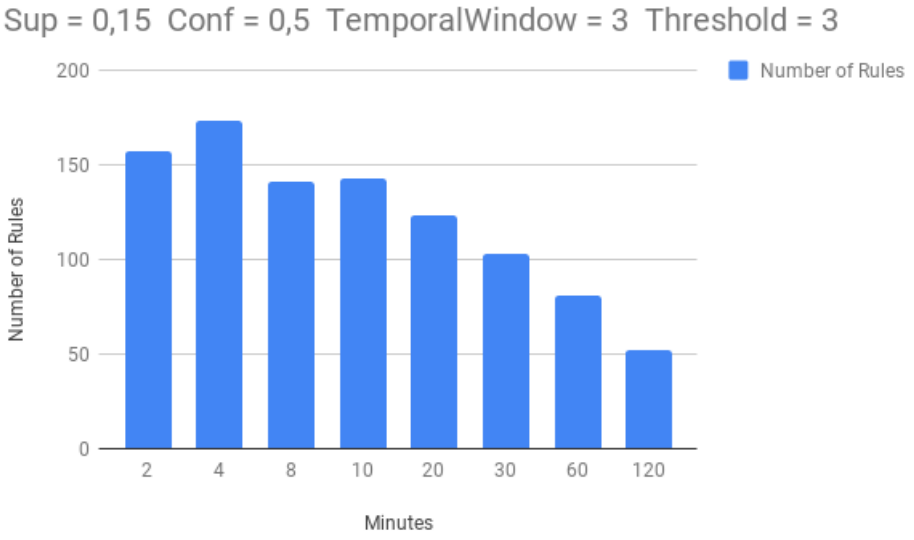

Figura 4.7. Relazione tra durata istante temporale e numero di regole estratte

Se la durata dell'istante temporale sembrava non avere alcuna relazione con la lunghezza media delle regole, sembra invece influenzare il numero di regole estratte. Come si può notare in figura, infatti, più aumenta il numero di minuti tra un istante temporale e l'altro e meno regole vengono generate in output dall'applicazione. Probabilmente questo risultato è dovuto al fatto che, su lunghi istanti temporali, alcune regole come quelle descritte nell'esperimento precedente non vengano considerate. Molte stazioni, infatti, pur essendo vicine geograficamente tra loro, a distanza di ore possono avere stati completamente differenti, mentre considerate in istanti temporali vicini e in determinati periodi del giorno (per esempio la notte o al mattino presto) possono avere lo stesso stato e creare più facilmente delle dipendenze e regole.

È inoltre interessante notare come in questo caso il numero di regole decresca in maniera meno immediata rispetto a come avviene per la confidenza e il supporto, non si ha infatti un decadimento esponenziale ma lineare, indice del fatto che la durata dell'istante temporale non rappresenta una soglia su cui le regole vengono filtrate ma influisce comunque sul risultato dell'estrazione.

4.3.7 Relazione tra valore di Threshold e lunghezza media delle regole

Uno degli aspetti più interessanti della fase di testing è stato osservare e verificare come il valore di threshold influisse sulle regole estratte dall'algoritmo. In prima analisi si sono effettuati dei test considerando il rapporto tra quest'ultimo e la lunghezza media delle regole, cercando di ipotizzare come il valore posto come soglia per stabilire lo stato di uno stazione potesse avere a che fare con i risultati. I valori utilizzati sono stati pochi, sia perché aumentando il valore di threshold aumentava in modo esponenziale il tempo di esecuzione dell'algoritmo (da pochi minuti fino a un'ora), sia perché, avendo ogni stazione un massimo di 18 postazioni, oltre un certo limite (posto a 8) non è sembrato sensato considerare una stazione empty o full. Per questo esperimento i valori di confidenza e supporto sono stati mantenuti a 0.5 e 0.15 come nei precedenti, il valore di threshold è stato fatto variare tra un minimo di 1 e un massimo di 8 mentre tutti gli altri parametri sono stati mantenuti ai valori degli esperimenti precedenti.

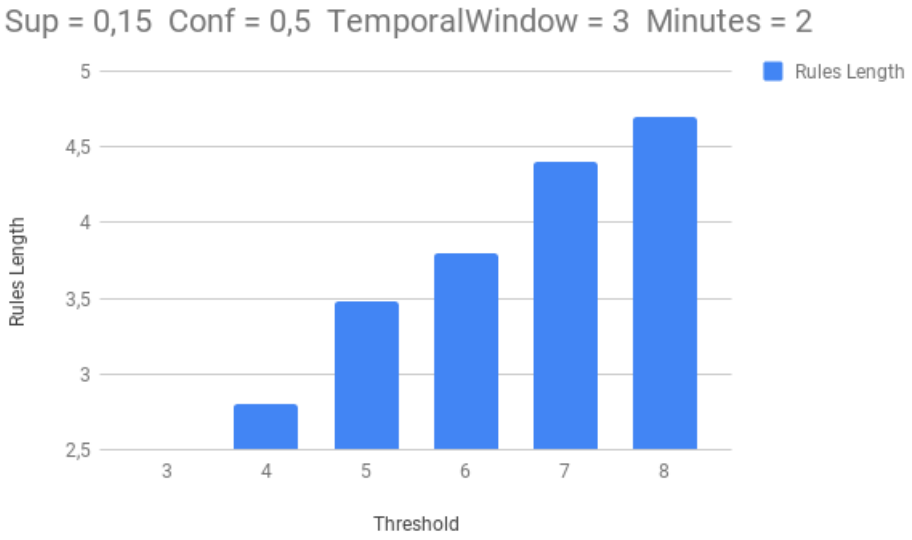

Figura 4.8. Relazione tra valore di threshold e lunghezza delle regole estratte

Tra tutti gli esperimenti questo è stato il più interessante e anche l'unico al quale è ancora difficile dare un'interpretazione. Al contrario di come è avvenuto nei test precedenti (e in particolare per il supporto), infatti, aumentando il valore di threshold aumenta la lunghezza media delle regole estratte. La crescita sembra essere di tipo lineare, anche se a seconda del valore di soglia posto la lunghezza media aumenta in maniera diversa.

4.3.8 Relazione tra valore di Threshold e numero di regole estratte

Il secondo esperimento effettuato sul valore di threshold ha riguardato la relazione tra quest'ultimo e il numero di regole estratte. Essendo il threshold un valore di soglia, si può ipotizzare che aumentandolo aumentino anche le stazioni in uno stato critico e dunque anche il numero di regole estratte. Supponendo infatti di usare come valore di soglia 5 rispetto a 3, si troverebbero in uno stato empty anche tutte le stazioni con 4 o 5 posti occupati e in uno stato full tutte quelle con 4 o 5 posti liberi. Per questo esperimento i valori utilizzati sono stati gli stessi del test precedente e anche in questo caso si è scelto di far variare poco la soglia per gli elevati tempi di esecuzione dell'algoritmo nel caso di alti valori di threshold.

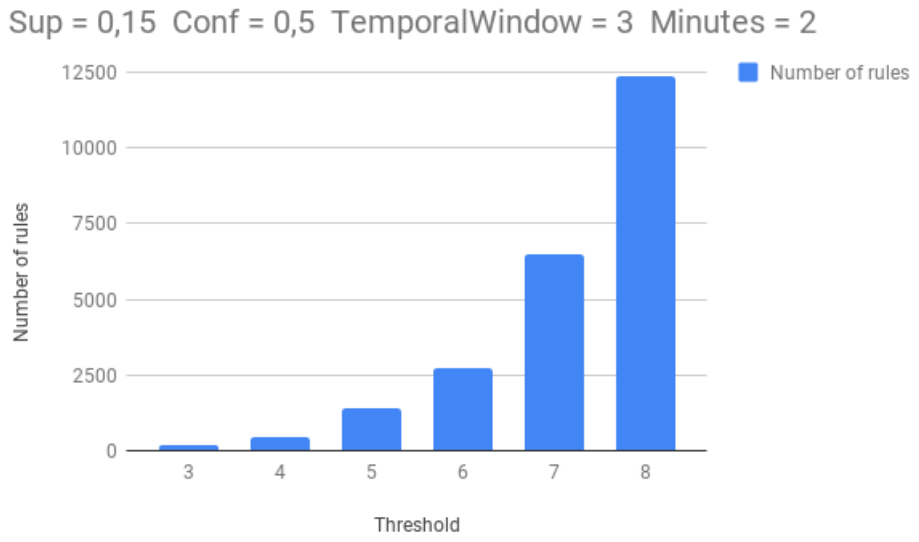

Figura 4.9. Relazione tra valore di threshold e numero di regole estratte

Come è possibile notare dal grafico le ipotesi sono state confermate; aumentando il valore di threshold aumenta anche il numero di regole estratte oltre che la loro lunghezza media. Questo perché aumentando la soglia entro la quale una stazione è considerata empty o full diventa più facile che una stazione abbia una criticità, aumenta il numero di record e istanti in cui una stazione si trova in uno stato critico, andando ad aumentare di conseguenza gli itemset e le regole in output all'applicazione. Dai risultati ottenuti si deduce che la crescita è di tipo esponenziale e questa dipendenza spiega anche il motivo per cui aumenta anche il tempo di esecuzione dell'algoritmo: essendo le regole in output salvate in un file di testo, maggiore è il numero di regole estratte e maggiore è il tempo impiegato per scriverle sul file.

4.3.9 Rapporto tra finestra temporale e confidenza media delle regole

Gli ultimi esperimenti e test relativi ai parametri dell'applicazione sono stati condotti andando a studiare come la variazione della finestra temporale incida sulle regole estratte. Tutte le regole vengono estratte considerano un certo numero di istanti temporali che definiscono una finestra all'interno della quale la singola regola è definita. La regola:

0_1_full, 1_2_empty -> 2_4_empty

ha finestra temporale pari a 3, perché considera gli istanti da 0 a 2, mentre la regola:

0_2_full -> 3_5_full

ha finestra temporale 4, perché nonostante abbia solo un antecedente e un conseguente considera gli istanti temporali da 0 a 3.

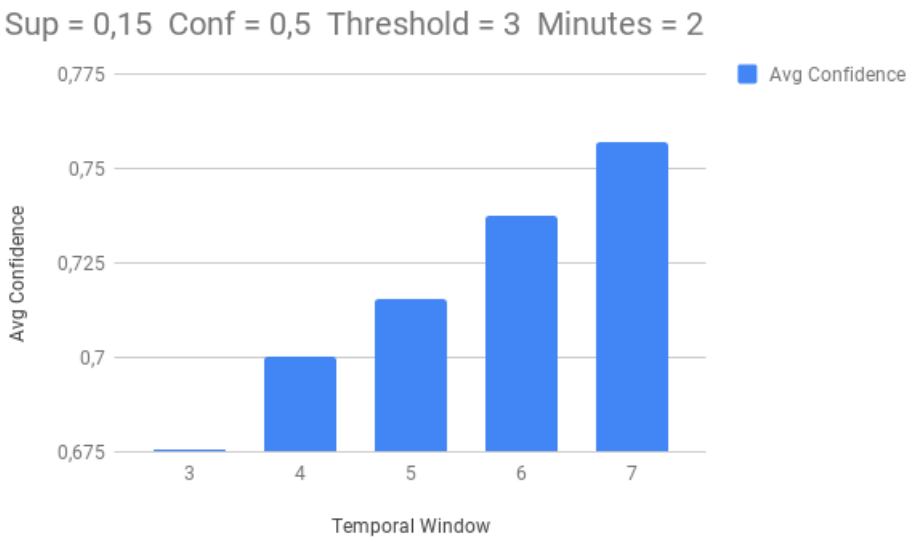

Figura 4.10. Relazione tra finestra temporale e confidenza media delle regole estratte

Il primo esperimento è stato condotto considerando la confidenza media delle regole estratte e, come si può notare dal grafico, essa è legata alla finestra temporale tramite una dipendenza lineare per cui aumentando il valore della seconda aumenta anche la prima. Questo risultato è interessante non solo per questo motivo, ma anche perché sembrerebbe indicare che, aumentando il numero di istanti temporali considerati, la precisione delle regole aumenti e l'algoritmo generi regole più affidabili e che si verificano più spesso.

4.3.10 Rapporto tra finestra temporale e numero di regole estratte

L'ultimo esperimento sui parametri dell'applicazione è stato fatto considerando il rapporto che intercorre tra la finestra temporale e il numero di regole estratte. Come visto nell'esperimento precedente, aumentando il numero di istanti temporali considerati aumenta anche la confidenza media delle regole considerate. La confidenza delle regole è però legata al loro numero, come è già stato discusso, tramite una dipendenza per cui maggiore è la prima e minore è il secondo. Queste considerazioni hanno portato all'interesse per questo esperimento, per cercare di capire quale tra valore della finestra temporale e confidenza media è in grado di influenzare maggiormente il numero di regole estratte.

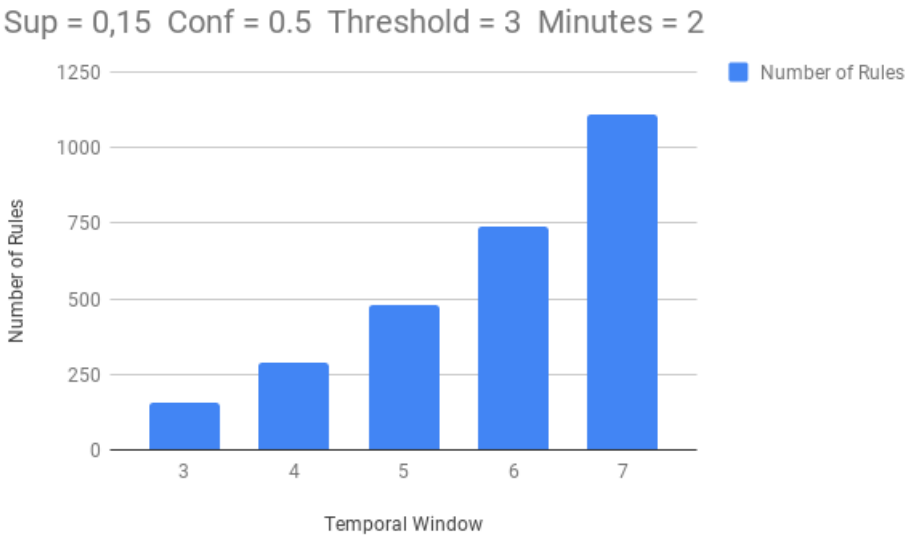

Figura 4.11. Relazione tra finestra temporale e numero di regole estratte

Il test ha rivelato, almeno per quanto riguarda questi parametri, che il numero di regole estratte è legato alla finestra temporale tramite un andamento di crescita che sembrerebbe di tipo esponenziale. Questo significa che considerando più istanti temporali vengono considerate più relazioni tra le stazioni in stato critico, probabilmente andando spesso a considerare le stesse stazioni in istanti temporali tra loro vicini.

4.4 Analisi predizione stati delle stazioni

La seconda tipologia di esperimenti descritti in questo capitolo riguarda l'efficienza dell'algoritmo proposto nel predire lo stato di una stazione in un determinato istante temporale e come essa varia a seconda dei parametri inseriti dall'utente. Per fare questo è stata utilizzata l'applicazione sviluppata come Visualizer, al cui interno è stato inserito del codice che, prima della visualizzazione delle regole, svolge le analisi. In questi test sono considerati tutti e tre gli stati possibili per una stazione (full, empty e normal) e, in particolare, vengono calcolati dall'applicazione:

1. FC: numero di stazioni predette full correttamente
2. EC: numero di stazioni predette empty correttamente
3. NC: numero di stazioni predette normal correttamente
4. FE: numero di stazioni full ma che sono state predette empty dall'algoritmo
5. FN: numero di stazioni full ma che sono state predette normal dall'algoritmo
6. EF: numero di stazioni empty ma che sono state predette full dall'algoritmo
7. EN: numero di stazioni empty ma che sono state predette normal dall'algoritmo
8. NF: numero di stazioni normal ma che sono state predette full dall'algoritmo
9. NE: numero di stazioni normal ma che sono state predette empty dall'algoritmo

Successivamente questi dati vengono utilizzati per calcolare la precisione delle predizioni, come vedremo in seguito. Tutte le analisi sono state fatte facendo variare la confidenza e la durata dell'istante temporale, mentre supporto, threshold e finestra temporale sono stati mantenuti fissi agli stessi valori in cui erano fissi negli esperimenti precedenti (Support = 0.15, Temporal Window = 3, Threshold = 3). In base al risultato dell'algoritmo di predizione descritto nel capitolo 3, i contatori vengono aggiornati. I risultati ottenuti sono mostrati nella seguente tabella:

Confidence	FC	FE	FN	EC	EF	EN	NC	NF	NE
0.25	135875	38708	86565	181297	23886	109309	286314	66015	117213
0.5	134114	34746	71007	176388	19706	83524	251268	61053	112303
0.75	75579	2631	16430	74627	2199	18293	101076	18501	22635
0.8	54446	1189	13824	32178	1505	13026	90813	10966	7338

Tabella 4.1. Risultati stazioni predette correttamente o in modo sbagliato in base alla confidenza

Come è possibile notare dai risultati, all'aumentare della confidenza il numero di predizioni corrette aumenta, mentre diminuiscono quelle sbagliate. Questo conferma il fatto che le regole con confidenza più alta sono più precise. Andando a considerare la precisione delle predizioni, ovvero calcolando il rapporto tra le predizioni corrette e quelle sbagliate si ottiene:

Confidence	FC/(FC+FW)	EC/(EC+EW)	NC(NC+NW)
0.25	0,520298834	0,576475713	0,609772928
0.5	0,559118178	0,630817759	0,591742341
0.75	0,798594675	0,784564598	0,71074171
0.8	0,783858103	0,688903637	0,832253453

Tabella 4.2. Variazione precisione delle regole in base alla confidenza

I valori relativi a FW, EW e NW rappresentano le predizioni sbagliate e sono stati calcolati nel modo seguente:

1. FW = FE + FN
2. EW = EF + EN
3. NW = NF + NE

Nonostante una piccola diminuzione per il valore di confidenza pari a 0.5 la precisione sembra aumentare insieme alla confidenza e, in particolare, si ha un picco per una confidenza pari a 0.75.

Dopo aver verificato come la precisione delle predizioni vari a seconda del parametro di confidenza inserito dall'utente è stato effettuato un test per vedere come essa è legata alla durata dell'istante temporale. In questo caso si è utilizzato un valore di confidenza fisso a 0.5 mentre tutti gli altri parametri (escluso il numero di minuti dell'istante temporale) sono stati mantenuti ai valori del test precedente.

Minutes	FC/(FC+FW)	EC/(EC+EW)	NC/(NC+NW)
20	0,587134294	0,561713904	0,595558906
30	0,589001895	0,567184796	0,588575583
60	0,606857211	0,56788355	0,559769793
90	0,523123457	0,541321616	0,594327861
120	0,510323854	0,539094888	0,581858798

Tabella 4.3. Variazione precisione delle regole in base alla durata dell'istante temporale con confidenza pari a 0.5

In questo caso i risultati sono di difficile intepretazione. Per valori bassi la precisione sembra aumentare (almeno per quanto riguarda le predizioni sulle stazioni di tipo full o empty) mentre oltre un certo limite essa cala drasticamente perdendo

anche, nel caso delle stazioni full, fino a 0.08 punti percentuale. Per quanto riguarda le stazioni normal, invece, tutti i valori sembrano aggirarsi intorno al valore ottenuto nel test precedente per una confidenza pari a 0.5. La durata dell'istante temporale sembrerebbe dunque non influenzare la precisione in una maniera definita o secondo qualche relazione stabilita.

Lo stesso test è stato ripetuto considerando una confidenza pari a 0.75.

Minutes	FC/(FC+FW)	EC/(EC+EW)	NC/(NC+NW)
20	0,81177246	0,727617947	0,752282876
30	0,764574364	0,76082904	0,80841929
60	0,906848125	0	0,866998593
90	0,937169283	0	0,871657452

Tabella 4.4. Variazione precisione delle regole in base alla durata dell'istante temporale con confidenza pari a 0.75

Per questo test oltre un certo numero di minuti, siccome la confidenza è stata posta pari a 0.75, l'algoritmo generava pochissime o addirittura nessuna regola. Questo è il motivo per cui nella tabella alcuni valori sono posti a 0 (l'estrazione con durata dell'istante temporale pari a 90 minuti ha generato solo poche regole e nessuna con un conseguente di tipo empty) e per cui si è scelto di non inserire i dati oltre a quelli ottenuti con un numero di minuti pari a 90.

Osservando questi risultati, a differenza di quelli con confidenza fissa a 0.5, la precisione delle regole sembra tendenzialmente aumentare man mano che aumenta anche la durata dell'istante temporale. In particolare, questa relazione è molto netta e visibile per le stazioni di tipo normal dove la precisione aumenta anche di 0.06 punti percentuale tra un risultato e il seguente.

Capitolo 5

Conclusioni

Gli esperimenti e le osservazioni effettuate hanno rivelato che l'algoritmo proposto è efficace e funziona nella misura dell'estrapolazione di regole e del loro utilizzo per prevedere lo stato di un evento in un luogo conoscendo un altro evento verificatosi in un altro luogo. Come detto nei capitoli precedenti, nonostante il metodo sia stato sviluppato e utilizzato per analizzare un caso studio particolare, esso è applicabile per qualsiasi esperimento in cui vengono presi in considerazione degli eventi georeferenziati. Sebbene i risultati siano stati positivi e abbiano rivelato come nel caso delle stazioni di Biciclette di Barcellona vengano raggiunti dei buoni valori di precisione relativi alle previsioni, il lavoro presentato può essere ampliato e migliorato con eventuali accorgimenti, alcuni dei quali vengono presi come esempio in questo capitolo.

Una delle ulteriori prove che sono state pensate e che per motivi di tempo non è stato possibile realizzare riguarda la necessità di ulteriori esperimenti su dataset diversi, per verificare come l'algoritmo si comporta effettivamente con dati diversi, pur essendo esso stato pensato per qualsiasi tipo di dati georeferenziati, e per stabilire se i risultati positivi ottenuti per il caso studio delle biciclette di Barcellona vengono confermati anche in altre situazioni. Questo è valido, in particolare, per la precisione ottenuta nelle predizioni delle regole, mentre per quello che concerne invece il rapporto tra i parametri dell'applicazione e le regole estratte bisogna considerare che, a seconda del dataset considerato, alcuni parametri potrebbero essere rimossi o aggiunti ad hoc (si pensi per esempio al parametro threshold nel caso studio delle stazioni di biciclette, esso non sarebbe utilizzabile per un dataset completamente diverso).

Un'altra idea che si potrebbe applicare per ampliare e migliorare il lavoro svolto riguarda il metodo di analisi ed estrazione delle regole e, in particolare, il modo in cui i dati vengono considerati prima che inizi la fase vera e propria di estrazione. Sarebbe infatti interessante applicare all'analisi dei dati una divisione spaziale simile a quella discussa nel capitolo 4 per i pattern frequenti nelle traiettorie, andando a divedere l'area geografica considerata in zone. Considerando l'esempio analizzato nel lavoro presentato, si potrebbe dividere l'area della città di Barcellona sulla base dei quartieri che la compongono ed estrarre, per ogni quartiere, delle regole a sè stanti. Questo potrebbe essere fatto applicando per esempio un filtro nella fase di preprocessing dei dati, considerando solo i record il cui id relativo al luogo è

contestualizzato all'interno di un'area geografica ben precisa e permetterebbe di studiare come sono correlate tra loro le regole relative alle stazioni presenti nello stesso quartiere.

L'integrazione più importante, tuttavia, riguarda la modalità di predizione del comportamento delle stazioni in un istante temporale. Come è stato descritto nei capitoli precedenti attualmente l'algoritmo effettua le predizioni considerando i dati e i record passati, andando a verificarle cercando nel file relativo agli eventi passati se alcuni pattern si sono verificati più volte in base alla regola considerata. Un'idea alternativa a questo metodo potrebbe essere quella di creare, grazie alle funzioni che la libreria MLLib di Spark mette a disposizione, un vero e proprio classificatore che prende in input i dati, elabora un modello e genera un training set (sul quale il classificatore viene allenato) e un test set (sul quale il classificatore viene testato). Una volta assegnate le label principali ai dati sarebbe possibile inserire qualsiasi altro dataset nel formato corretto in input e il classificatore genererebbe automaticamente delle regole di associazione simili a quelle ottenute come risultato nel caso studio preso in esempio.

Oltre che per la parte dell'applicazione dedita al processamento dei dati e all'estrazione delle regole, è possibile pensare a delle migliorie e delle estensioni anche per la parte di visualizzazione dei risultati. Innanzitutto si potrebbe considerare un numero maggiore di regole per quanto riguarda le predizioni e dunque le stazioni visualizzate dall'applicazione grafica. Per come è stata sviluppata l'applicazione, infatti, la precisione delle predizioni viene calcolata considerando solo la regola con confidenza più alta per ogni stazione. Essendo le regole inserite in una struttura dati ordinata per confidenza questa è una scelta che permette di rendere più veloce ed efficiente l'algoritmo. Considerando però tutte le regole inserite nella struttura dati aumenterebbe il numero di prove su cui viene verificato se l'algoritmo ha effettuato una previsione corretta e la precisione ottenuta in questo mondo sarebbe più attendibile.

Un'altra estensione della parte grafica potrebbe riguardare la risoluzione di alcune problematiche dovute alla libreria GMapsFX utilizzata che, sebbene permetta in modo semplice e immediato di interagire con una mappa geografica uguale in tutto e per tutto a quella fornita dall'applicativo di Google Maps, consiste in un wrapper delle API Javascript ufficiali, è open source e ancora contiene molti bug e carenze che saranno sicuramente implementate in futuro. Un esempio è dato dall'impossibilità di inserire, in corrispondenza dei marker presenti sulla mappa, una descrizione vera e propria; quello che la libreria concede è solo di inserire una parola al suo interno, feature che è stata utilizzata per inserire i numeri identificativi delle stazioni nelle loro posizioni geografiche. Questo non ha permesso anche di inserire in corrispondenza delle stazioni il relativo stato e la regola generale a cui esse fanno riferimento, che per questo motivo è stata spostata graficamente in alto in una parte superiore della visuale grafica proposta all'utente.

Un altro esempio si ha invece relativamente alle icone utilizzate per rappresentare i marker che identificano le stazioni all'interno dell'applicazione grafica. Attualmente sono previste, infatti, le classiche icone di Google Maps monocromatiche che sono visibili nell'immagine seguente:

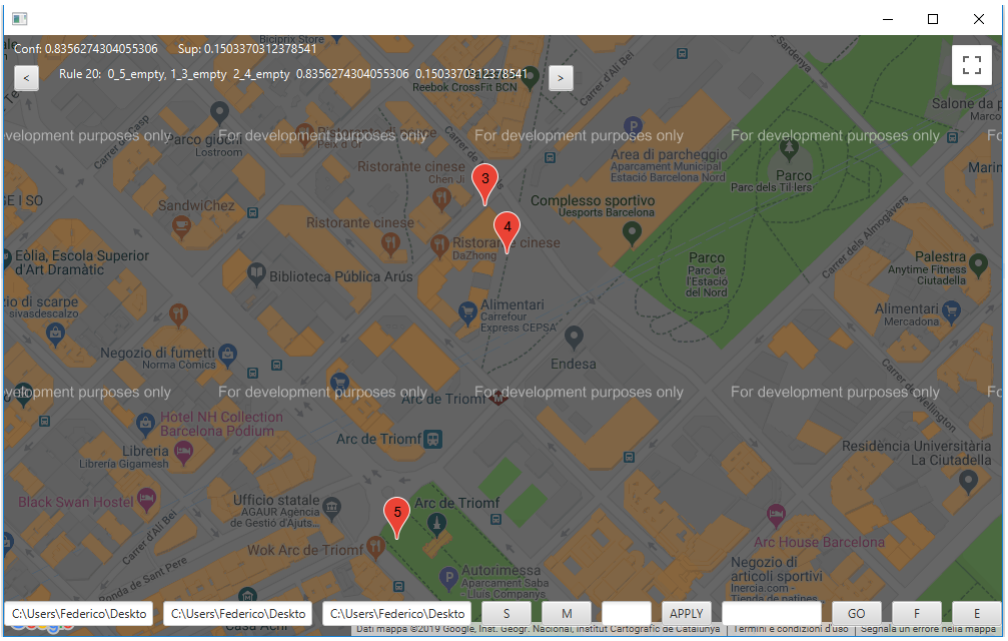

Figura 5.1. Esempio di marker utilizzati nell'applicazione grafica

La scelta di utilizzare queste icone è data dal compromesso ottenuto con l'utilizzo della libreria GMapsFX che, per via di alcuni bug non ancora risolti e anche se prevede al proprio interno la possibilità di inserire un'immagine custom per la rappresentazione delle icone, non permette attualmente di settare un'icona scelta dall'utente. Questo sarebbe molto utile, per esempio, per differenziare i luoghi rappresentati sulla mappa in base allo stato dell'evento a cui fanno riferimento; nel caso dell'esempio studiato nel lavoro di tesi sarebbe possibile rappresentare graficamente in modo diverso le stazioni in stato full da quelle in stato empty e questo permetterebbe ad un eventuale utilizzatore dell'applicazione di capire immediatamente e senza troppo sforzo in quali stati si trovano le stazioni che caratterizzano la regola, rendendo il tutto più user-friendly.

Nonostante queste integrazioni applicabili sia alla parte di processing dei dati che a quella di visualizzazione dei risultati, l'algoritmo proposto nel lavoro di tesi ha riscontrato dei buoni risultati e rappresenta un buon punto di partenza per poter studiare e analizzare un dataset di elementi georeferenziati, in particolare le motivazioni per cui scegliere l'approccio proposto sono molteplici. Innanzitutto la generalizzazione che l'algoritmo propone permette di far sì che esso sia applicabile a un numero di contesti variegato e molto interessante, senza dipendere troppo dal tipo di dati (come visto, viene effettuata la trasformazione dei dati in input in un formato standard che l'algoritmo riconosce), inoltre tutte le trasformazioni che lo compongono sono trasformazioni semplici (filtraggio, ordinamento e mapping) che possono essere applicate da qualsiasi engine adatto all'analisi dati, senza doversi per forza legare quindi, per esempio, ad Apache Spark. La parte di visualizzazione dei dati, allo stesso modo, consente una visualizzazione di qualsiasi tipo di dato espresso

nel formato elaborato dall'algoritmo, facendo così della flessibilità il punto forte e chiave di questo approccio.

Il lavoro presentato nel documento è solo uno degli esempi di studi che sono stati svolti e che continuano a essere svolti sull'analisi di dati spazio-temporali e, più in generale, sull'intero mondo dei Big Data. Come già detto nel capitolo 2, la ricerca e il mercato stanno investendo molto su questo ambito, finanziando esperimenti e cercando di integrare nel loro business gli strumenti più recenti e più performanti. Il campo dell'analisi dati rappresenta, insieme a quello della cyber-security, sicuramente uno degli ambiti più interessanti e che più sarà richiesto in futuro, complice il fatto che viviamo in un mondo sempre più intriso in un continuo flusso di dati che, senza gli strumenti adatti, diventa molto difficile interpretare. Proprio per questo motivo uno studio di questo tipo è utile e permette di cominciare ad aprire delle strade, delle formalizzazioni e delle metodologie per riuscire ad adattare, sempre di più, la quantità enorme di informazioni che ci circonda alle nostre esigenze.

Bibliografia

[1] G. Atluri, A. Karpatne, V. Kumar *Spatio-Temporal Data Mining: A Survey of Problems and Methods*, ACM Computing Surveys, Vol. 51, No. 4, Article 83, 2018.

[2] J. Mennis, J. Wei Liu *Mining Association Rules in SpatioTemporal Data: An Analysis of Urban Socioeconomic and Land Cover Change*, 2005.

[3] P. Compiete, S. Di Martino, M. Bertolotto, F. Ferrucci, T. Kechadi *Exploratory spatio-temporal data mining and visualization*, Journal of Visual Languages and Computing, 2007

[4] http://dbdmg.polito.it/wordpress/, sito ufficiale del gruppo database and mining del Politecnico di Torino

[5] https://spark.apache.org/, sito ufficiale di Apache Spark

[6] https://openjfx.io/, sito ufficiale JavaFX

[7] https://docs.oracle.com/javase/8/docs/api/, documentazione ufficiale Java SE 8

[8] https://developers.google.com/kml/documentation/, documentazione ufficiale per il linguaggio KML (Keyhole Markup Language

[9] https://gluonhq.com/products/scene-builder/, sito ufficiale plugin JavaFX SceneBuilder

[10] https://www.google.it/intl/it/earth/, sito ufficiale Google Earth, tool online per la visualizzazione di informazioni spaziali

[11] https://maps.google.com/, sito ufficiale Google Maps, tool online per la visualizzazione di informazioni spaziali

[12] https://rterp.github.io/GMapsFX/, repository ufficile della libreria open source GMapsFX, utilizzata per la realizzazione dell'applicazione grafica relativa alla visualizzazione dei risultati

[13] http://www.eclipse.org/efxclipse/, tool e plugin di Eclipse che permette l'integrazione con JavaFX e la possibilità di usare le funzioni native presenti al suo interno